ONOS
Open Network Operating System

Andrea Campanella, ONF, USA
Abdulhalim Dandoush, ESME-SUDRIA, France
Lefteris Manassakis, ICS FORTH, Greece
ONOS Build, SAMSUNG Seoul
September 20, 2017
General Information

WIFI
SSID: onos_creative_<number>
PWD: onosbuild2017

Charge laptops downstairs if you need
Links and Virtual Machine

Introduction: https://goo.gl/Vk1gjh
Installation: https://goo.gl/1BjLWa
Apps: https://goo.gl/fB5yfJ
VM : https://goo.gl/i1d9Rs
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.

Quarterly Releases, **Loon** (1.11.0) - released 2017-10
Service Provider Networks

● **WAN core backbone**
  o Multi-Protocol Label Switching (MPLS) with Traffic Engineering (TE)
  o 200-500 routers, 5-10K ports

● **Metro Networks**
  o Metro cores for access networks
  o 10-50K routers, 2-3M ports

● **Cellular Access Networks**
  o LTE for a metro area
  o 20-100K devices, 100K-100M ports

● **Wired access / aggregation**
  o Access network for homes; DSL/Cable
  o 10-50K devices, 100K-1M ports
Architectural Tenets/principles

- High-availability, scalability and performance
- Strong abstractions and simplicity to develop apps and solutions
- Protocol and device behaviour independence
- Separation of concerns and modularity
ONOS applications

ONOS networking core

ONOS distributed applications platform

OSGI / Apache Karaf
Distributed Core
ONOS Distributed Architecture

- **Distributed**
  - Set up as a cluster of instances

- **Symmetric**
  - Each instance runs identical software and configuration

- **Fault-tolerant**
  - Cluster remains operational in the face of node failures

- **Location Transparent**
  - A client can interact with any instance. The cluster presents the abstraction of a single logical instance

- **Dynamic**
  - The cluster can be scaled up/down to meet usage demands
tcp:9876
ONOS Cluster
Closer look...
State Management in ONOS

- Better if applications can focus on business logic
- Primitives span the consistency continuum
  - `EventuallyConsistentMap<K, V>`
    - Map abstraction with eventual consistency guarantee
  - `ConsistentMap<K, V>`
    - Map abstraction with strong linearizable consistency
  - `LeadershipService`
    - Distributed Locking primitive
  - `DistributedQueue<E>`
    - Distributed FIFO queue with long poll support
  - `DistributedSet<E>`
    - Distributed collection of unique elements
Northbound

intents resources network config

network-centric
Interact with ONOS: GUI

UI: <onos-ip>:8181/onos/ui
Interact with ONOS: CLI

$onos <controller_address>
Interact with ONOS: REST and GRPC

REST APIs: <onos-ip>:8181/onos/v1/docs/

<table>
<thead>
<tr>
<th>flows</th>
<th>Query and program flow rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELETE</td>
<td>/flows/application/(appId)</td>
</tr>
<tr>
<td>GET</td>
<td>/flows/application/(appId)</td>
</tr>
<tr>
<td>DELETE</td>
<td>/flows</td>
</tr>
<tr>
<td>GET</td>
<td>/flows</td>
</tr>
<tr>
<td>POST</td>
<td>/flows</td>
</tr>
<tr>
<td>DELETE</td>
<td>/flows/(deviceID)/(flowID)</td>
</tr>
<tr>
<td>GET</td>
<td>/flows/(deviceID)/(flowID)</td>
</tr>
<tr>
<td>GET</td>
<td>/flows/(deviceID)</td>
</tr>
<tr>
<td>POST</td>
<td>/flows/(deviceID)</td>
</tr>
</tbody>
</table>

Northbound GRPC with protocol buffers (.proto) for ONOS network model
Interact with ONOS

gRPC:
- Google remote procedure call
- Based on protobuf
- Faster than REST
- Off-Platform applications

RESTCONF
- Rest interface for the XML-based rpc and streaming operations
Key Northbound Abstractions

- **Network Graph**
  - Directed, cyclic graph comprising of infrastructure devices, infrastructure links and end-station hosts

- **Flow Objective**
  - Device-centric abstraction for programming data-plane flows in table pipeline-independent manner

- **Intent**
  - Network-centric abstraction for programming data-plane in topology-independent manner
Network Programming

Abstract to concrete

Intent

Host-Host
Single-Point to Multi-point
Protected Intent

Flow Objective

OFDPA Pipeline
Single Table Pipeline
P4 program Defined Pipeline

Flow Rule

Mapping through drivers

OF
P4Runtime
Netconf
TL1
Flow Objectives
Flow Objective Architecture

Device driver is used to translate objectives to the specific flow rules for a given device and pipeline.

FlowObjective Service

Pipeliner API

- CorsaPipeline
  - OFFLowRuleProvider
    - Open Flow 1.0 Rule
    - Corsa Pipeline

- Juniper Pipeliner
  - OFFLowRuleProvider
    - Netconf XML
    - Juniper FlowRuleProgrammable
    - Juniper Pipeline

- OFDPA Pipeliner
  - OFFLowRuleProvider
    - Open Flow 1.3 Rule
    - HP Pipeline
Flow Objective example

Peering Router Match on Switch port, MAC address, VLAN, IP
Flow Objective Summary

- *Flow Objective:* abstraction for applications to be pipeline unaware while benefiting from multi-table architectures

- Enable app portability
  - interoperability between different types of pipelines coexisting in heterogeneous networks.
  - Support for a new pipeline is achieved through a new pipeliner behaviour in a driver. The new pipeline is then programmable from all existing applications.
Intent Framework

• Provides high-level, network-centric interface that focuses on what should be done rather than how it is specifically programmed

• Abstracts unnecessary network complexity from applications

• Maintains requested semantics as network changes

• High availability, scalability and high performance
Intent Example

Application

Java, Rest, gRPC

CREATION

Host to Host Intent
Intent Example

Host to Host Intent

submit()

Intent Service API
Intent Example

Host to Host Intent

Path Intent

Path Intent
Intent Example

Host to Host Intent

Path Intent

Flow Rule Batch

Path Intent

Flow Rule Batch

Flow Rule Batch

Flow Rule Batch

Flow Rule Batch

COMPILATION

INSTALLATION
Intent Framework Summary

- Intents are a network-centric programming abstraction that reduce application complexity.
- Intents provide device-agnostic behavior with persistency and high performance across network failures.
- Intent framework has moved from prototype to production deployments.
Failures in Intents
Reaction to failures for intents

All the intents that are involved in the failure will be **re-compiled** with a multi-threading process that:

1. **removes** the old failed intents
2. finds a **new entire path** from source to destination
3. generates **new Installable Intents** *(i.e., FlowRule/FlowObjective Intent)*
Configuration

- Network Configuration (netcfg)
  - Provides mechanism for any service to register and receive configuration

- Device Configuration
  - Behaviors abstract the management and configuration aspects of a device

- Dynamic Configuration
  - Enables YANG-based service models to be introduced at runtime
  - Allow applications to implement dynamic services
Dynamic Config
Control and Configuration

- Operators need a resilient and scalable platform capable of *both* control and configuration.
Dynamic Configuration

- *.yang
  - YANG Compiler
  - model.jar
  - YANG Runtime
  - model.jar
  - *.yang

- Device Certification App
- Device Config App
- Network Function
- Dynamic Config Subsystem
- REST / gRPC / RESTCONF / NETCONF NB
- RESTCONF / NETCONF SB
- Distributed Config Store
- /devices
- /services
- JSON / XML
- JSON / XML
- Device
- Device
- Device
Southbound
Southbound overview

Southbound protocols in 1.11.0:

- OpenFlow until 1.3 + optical extension, 1.5 is implemented and under testing
- OVSDB
- NETCONF + YANG → Yang tools and Yang management system
- SNMP
- P4 → thrift api for bmv2 softswitch from barefoot networks. → p4Runtime
- BGP, ISIS, OSPF → interoperability with legacy network.
- PCEP → Path computation element protocol (IETF)
- REST and RESTCONF
- LISP
- TL1
- gRPC
Providers, protocols and drivers

- ONOS interacts with the underlying network with the help of its Providers.

- **Providers**: hide complexity from upper layers, used by the core to (re)act on the network

- **Protocols**: Features and modules to communicate with devices

- **Drivers**: define specific capabilities offered by the device
ONOS Protocols

- Features and modules to communicate with devices
- Expose the standard set of APIs and enabled operations. I.e:
  - OpenFlow: FlowMods, GroupMods, etc
  - Rest: implements CRUD operations (GET, POST, DELETE, etc...)
  - Netconf: Open/close session, setConfiguration, getConfiguration
- Usually leverage 3rd party communication libraries → openflowj, snmp4j, thrift
ONOS Providers

Providers are used by the core to (re)act on the network:

- Up/down of device, links
  - DeviceProvider, LinkProvider
- Provisioning of rules, paths, tunnels
  - FlowRuleProvider, TunnelProvider
- Process packets, notifications/alarms
  - PacketProvider, AlarmProvider

→ Translate to and from Core abstractions into protocol or behaviours calls.
ONOS drivers

- Device specific driver
  - collection of behaviors
  - on-demand activation

- Abstraction via behaviors
  - define specific capabilities offered by the device
  - encapsulate specific logic and code
    - ports, controller, flowrule, power…

- Encapsulate single interaction
  - protocol
  - information

```
<driver name="default" manufacturer="ON.Lab"
  hwVersion="0.0.1" swVersion="0.0.1">
  <behaviour api=InterfacePath
    impl=ImplementationPath />
</driver>
```
ONOS CORE Behaviours

<behaviour api=InterfacePath
impl=ImpementationPath />

APIs

- **DeviceDescriptionDiscovery** → Device’s Description and Ports.
- **FlowRuleProgrammable** → Translates to and from ONOS core FR abstraction in device specific calls
- **PacketProgrammable** → Emit a given packet from a device.
- **GroupProgrammable** → Translates to and from ONOS Groups
- **PortStatisticsDiscovery** → Statistics of Device Ports.
- **Pipeliner** → Pipeline abstraction, FlowObjectives to pipeline specific FR

Recently Introduced

- **DeviceHandshaker** → device handshake for GeneralDeviceProvider.
- **PipelineProgrammable** → Installs a Programmable Pipeline on the Device.
Example: FlowRuleProgrammable

FlowRuleManager

Apply Flow rule request

OpenFlowRuleProvider

of:0000000000000001

FlowRuleDriverProvider

netconf:foo@1.2.3.4:567

FooFlowRuleProgrammable

DriverService

driver: foo-netconf
behavior:
interface: FlowRuleProgrammable
impl: FooFlowRuleBehavior

OpenFlow

1.2.3.4:567

NETCONF

Southbound

Application or ONOS subsystem
Applications
Developing ONOS applications

ONOS applications:

- Interact with the northbound Java, REST, gRPC interface
- **Device and protocol agnostic**
- **Augment ONOS** though modularity
- Provide GUI, REST, CLI and distributed stores.
- Shape the network.
- Easy to start with **auto generated basic code** via maven archetypes.
Example Applications

● SDN-IP Peering
  ○ Connect internal BGP software daemon to external BGP routers
  ○ Install learned routes to forward IP traffic to appropriate egress point

● Multi-level (IP / Optical) Provisioning
  ○ Provision optical paths/tunnels with constraints

● Content Acquisition / Video Streaming (DirecTV)
  ○ Establish multicast forwarding from a sender to set of receivers

● Virtual Network Gateway (vBNG)
  ○ Provide connectivity between a private host and the Internet

● Bandwidth Calendaring
  ○ Establish tunnels with bandwidth guarantees between two points at a given time
ONOS NREN Deployments

• Expand
  • Deploy ONOS and its apps on your network

• Guide
  • ONF provides resources, guides and tips to help your deployment

• Share
  • ONF shares your deployment with the world

Q3 2015 – New connections
Sidney – Seattle - Miami
Sao Paolo – Amsterdam

Q4 2015 – First ONOS production deployment in South America

Q1-Q2 2015 First ONOS Deployments
South America, US, EU

Q1 2016 – New connections
Miami - Korea
Korea - Taiwan

Q3 2015
Korea announces the first ONOS deployment
Q1 2016
NCTU / Taiwan deploys ONOS

Q3 2015
ONOS deployed in Australia

eBGP over L2 dedicated circuits
System Requirements for ONOS

Hardware requirements are difficult to define. They depend on:

- Cluster size (i.e., # of instances),
- Managed network size (e.g., devices, links, ports, etc.)
- Number of messages exchanged with network devices
- Number of flows

**Basic requirements** for learning/testing

- 2 core CPU
- 2~4 GB RAM
- 10 GB hdd
- 1 NIC (any speed)
Andrea: andrea@opennetworking.org
Lefteris: manassakis@gmail.com
Abdulhalim: adandoush@gmail.com
Further reading

ONOS website:
http://onosproject.org
Tutorials, documentation and general reading at:
https://wiki.onosproject.org/
ONOS is on Github at:
https://github.com/opennetworkinglab/onos
Setup Tutorial
https://wiki.onosproject.org/display/ONOS/Installing+and+Running+ONOS
Screencasts:
https://wiki.onosproject.org/display/ONOS/Screencasts
Software Defined Transformation of Service Provider Networks

Join the journey @ onosproject.org
ONOS and P4: Runtime control
Today: P4Runtime

● Program-independent runtime API for P4-enabled devices
  ○ gRPC-based, p4runtime.proto protobuf definition

● Allow interacting with entities defined in a P4 program
  ○ Tables, counters, meters, externs, etc.

● Independent from the actual instance of the P4 program
  ○ No code auto-generation

● Specification is still work-in-progress by API WG
P4Runtime API in a nutshell

- **Write messages**
  - Add/modify/remove entities (e.g. table entries, groups, etc.)
  - Blocking calls

- **Read messages**
  - Read content of entities (e.g. to retrieve all entries of a table)
  - Non-blocking (async)

- **Allow to set/get forwarding pipeline**
  - Based on `p4info.proto` definition of the pipeline

- **Bidirectional stream channel**
  - For packet-ins, packet-outs, mastership notifications
Overview

2 modes of operations:
- Standard criteria, treatments
- Pipeline-specific ones

FlowObjectives, Intents

Events (packet, topology, etc.)

FlowRules

Pipeline-agnostic applications

Pipeline-aware application

Core

Driver

Protocol

Device

(BMv2, Tofino, etc.)

PI Flow Rule Translation Serv.

PI Pipeconf Serv.

Tofino

BMv2

Default

P4Runtime vX

gNMI vX

Protobuf, HTTP/2

PI Pipeconf (.oar)

new
Flow Objective Abstraction

- **Problem:** Applications currently must be pipeline aware, effectively making applicable on specific HW. **Flow objectives enable developers to write applications once for all pipelines**

First attempt at interoperability between OF 1.3 switch

Controller

Write the app once without pipeline details

SDN Application

Flow Objectives

Driver X

Driver Y

Driver Z

Switch

adapt to specific Pipeline using Switch driver

Pipeline X

Pipeline Y

Pipeline Z

OpenFlow 1.3

OpenFlow 1.3
Flow rule provider architecture

Flow Rule Driver Provider

Get/Apply/Remove FlowRule

P4 Runtime Flow Rule Programmable Impl

translate(FlowRule, Device)

gRPC controller

Southbound

Pi Pipeconf Manager

cget PipelineConfiguration(Device)

Core