OpenStack and OpenDaylight: The Way Forward

Colin Dixon, Flavio Fernandes, Kyle Mestery
Some content from: Sam Hague, Anil Vishnoi, David Meyer, and Neela Jaques
What’s coming

• What is OpenDaylight?
• OpenStack Support in OpenDaylight
• Demo
• Future Plans
• Questions
What is OpenDaylight

OpenDaylight is an **Open Source Software** project under the **Linux Foundation** with the goal of furthering the adoption and innovation of **Software Defined Networking (SDN)** through the creation of a common industry supported platform.

**Code**

To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution.

**Acceptance**

To get broad industry acceptance amongst vendors and users:
- Using it directly or through vendor products
- Vendors using OpenDaylight in commercial products

**Community**

To have a thriving and growing technical community contributing to the code base, using the code in commercial products, and adding value above, below and around.
What makes OpenDaylight different?

• Inclusive view of SDN
  • Support for SNMP, BGP, PCEP, COPS/PCMM, LISP, NETCONF, OpFlex…
  • …and OpenFlow and OVSDB

• Broad, engaged-community
  • 3200 commits from 150 contributors in the last 30 days

• In other words, it’s a lot like OpenStack
OpenDaylight Releases

• **Hydrogen** (first release)
  • February 2014
  • 13 projects, 1.3m lines of code

• **Helium** (most recent release)
  • October 2014
  • 25 projects, 2.1m lines of code

• **Lithium** (upcoming release)
  • Planned June 2015
  • 40+ projects, 2.3m lines of code
Who is OpenDaylight?
Who is OpenDaylight? (Really)
Who is OpenDaylight? (Really)

• Like any Open Source Project, OpenDaylight primarily consists of those who show up to do the work.

• Running around 300 commits per week over 12 months, trending up
  • 30 Days: ~3200 commits, ~150 contributors (4/1/15–5/1/15; during a release)
  • 12 Months: ~16,000 commits, ~325 contributors (5/1/14–5/1/15)

• Strong integration and testing community
  • This stuff really matters

Source: https://www.openhub.net/p/opendaylight
OpenStack Support in OpenDaylight
An OpenStack view of OpenDaylight

- OpenDaylight has a common Neutron “northbound” provider
  - 3 implementations in Helium
    - OVSDB, OpenContrail, VTN
  - 5+ implementations in Lithium
    - OVSDB, VTN, LISP, Group-based Policy, VPN service

- Supports ML2 and some advanced services
Demo

OpenStack with OpenDaylight (L2+L3 east-west)
An OpenStack view of OpenDaylight
Lithium adds YANG and more providers

OpenStack Neutron

- Neutron ML2 MechanismDriver
- OpenDaylight APIs (REST)
- Neutron Service
- OVSDB Provider
- VTN Provider
- OpenContrail Provider

OpenDaylight

OpenStack Neutron

- Neutron ML2 MechanismDriver
- Neutron REST API
- OpenDaylight YANG-modeled RESTCONF
- Neutron Service
- VTN Provider
- VTN Provider
- Group Policy Provider
- LISP Provider
- VPN Service Provider

OpenDaylight
Disaggregating the OVSDDB monolith

Helium
- OpenStack
  - Neutron (REST)

OVSDDB
This project is a monolithic combination of:
(1) a network virtualization layer that is "hard-wired" to Neutron above and OVS below as well as (2) an OVSDDB protocol library.

- OpenFlow + OVSDDB + Nicira Extensions

Lithium
- OpenStack
  - Neutron (REST)
  - Neutron (YANG)

OVSDDB Network Virtualization
Network virtualization layer that is still "hard-wired" to Neutron above, but now uses more general APIs below.

- REST/YANG Adapter
  - Tunnel Mgmt
  - Traffic Direction
  - OVSDDB Plugin
  - OpenFlow + Nicira Extens?

Future
- OpenStack
  - Neutron (REST)
  - Neutron (YANG)

OVSDDB Network Virtualization
Network virtualization layer that now uses the more general APIs above and below.

- REST/YANG Adapter
  - Tunnel Mgmt
  - Traffic Direction
  - OVSDDB Plugin
  - OpenFlow + Nicira Extens?
  - NETCONF
  - Many h/w- and v-switches
Beyond Lithium

- Whatever you want us to do! Tell us what matters most to you.
  - ...and show up and help do the work!

- Planned activities:
  - More advanced services
  - Stability, scalability, performance, ease of deployment
  - Test, test, and more testing: it turns out the tests are more valuable than the code

- If there’s interest
  - Added southbound support, e.g., NETCONF tunnel mgmt
Questions?

https://wiki.opendaylight.org/view/Main_Page
https://wiki.opendaylight.org/view/OpenStack_and_OpenDaylight
Backup Slides
Core Architecture

Model-Driven Service Abstraction Layer (MD-SAL)
- Notifications
- RPCs

Data

Controllers in a Cluster

App/Service

Plugin

YANG Models

OPEN DAYLIGHT
Yang Models

container network-topology {
  ...

  list node {
    description "...";
    key "node-id";
    uses node-attributes;
  }

  list link {
    description "...";
    key "link-id";
    uses link-attributes;
  }
}

- Network Topology
- List of Nodes
- List of Links
- Links and Nodes can be “augmented” later

- Can have multiple topologies
  - Overlay/underlay
  - Disjoint
  - Peered
SDN Grand Challenges

• Centralized vs. Distributed
  • RAFT distributed consensus algorithm in Helium
  • Continued work on clustering in Lithium and beyond

• Migration to SDN
  • Support SNMP, BGP, LISP, NETCONF “legacy” protocols

• Application Composition
  • Support for declarative, intent-based policy
  • Unified models for inventory, topology, and more

• Hardware Diversity
  • Support for Table Type Patterns
  • Device Driver Framework will provide adaptation in Lithium
How to get there from here

• How do we deploy SDN when it’s not green field
  • Because pretty much nothing is actually green field
  • Hybrid switches, hybrid networks, legacy protocols for interoperability, etc.
    • OpenDaylight supports SNMP, BGP, LISP, NETCONF, etc.

• Trust and stability
  • Current networks build on 40 years of code/experience
  • How can SDN compete with that?
    • Borrow good code/ideas from legacy code
    • Provide better visibility, debugging, etc.
    • Model checking, verification, etc.
Centralized vs. Distributed  
(Consistency, Clustering and Federation)

• SDN promises a (logically) centralized control plane
• In practice, we have a distributed cluster of controllers, rather than just one so that
  • we can tolerate faults
  • we can scale out our performance
  • in network partitions there are controllers on both sides
• Providing consistency, federation, scale-out, dealing with CAP trade-offs, etc. is HARD
Hardware Diversity

• OpenFlow 1.0 provided a lowest common denominator API
  • Real hardware is much more diverse
  • and has many more capabilities

• Exposing this diversity without burdening developers with per-device programming is hard

• Some Attempts
  • Programming Protocol-Independent Packet Processors
  • TTPs from the ONF’s FAWG
Application Composition

• How can we let multiple SDN apps share the network?
  • PC OSes partition and allocate resources
  • You can’t easily partition the network
    • It’s value comes from the fact that it spans everything
    • You can in some cases, e.g., by address space (FlowVisor)

• Some ideas
  • Most apps should be middleboxes, i.e., NFV
    • Simply chain them together in the right order
    • There’s more to it than this, but linear chaining is powerful
  • Other apps are concerned only with the physical path
    • There is hope that conflicts here can be sanely managed