OpenDaylight and the Open Source Future of Networking

Colin Dixon, Principal Engineer, Brocade

colin@colindixon.com  (http://colindixon.com)
@colin_dixon

Some content borrowed from David Meyer, Kyle Mestery, Anees Shaikh, and Luis Gomez
What is OpenDaylight

Some things I’ve learned

A Few OpenDaylight Metrics

Where OpenDaylight is Going

SDN Grand Challenges
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.

<table>
<thead>
<tr>
<th>Code</th>
<th>Acceptance</th>
<th>Community</th>
</tr>
</thead>
</table>
| To create a robust, extensible, open source code base that covers the major common components required to build an SDN solution | To get broad industry acceptance amongst vendors and users  
  - Using OpenDaylight code directly or through vendor products  
  - Vendors using OpenDaylight code as part of commercial products | 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 is OpenDaylight building?

- An evolvable SDN platform capable of handling diverse use cases and implementation approaches
  - Common abstractions for people to program
    - “Northbound” Interfaces
  - Southbound “drivers”, e.g., OpenFlow, OVSDB, BGP-LS
  - Intermediation between north and south
- Programmable Network services
- Network Applications
- Whatever else we need to make it work
Project Framework

- Network applications, orchestration, and services
  - User interfaces
  - Network applications, orchestration, and services

- OpenDaylight APIs (REST)
  - Network service functions
  - Platform services
  - Extensions

- Controller platform
  - Southbound interfaces & protocols
  - OpenFlow
    - Other standard protocols (ONF, IETF, ...)
    - Vendor-specific interfaces

- Data plane elements (virtual switches, physical device interfaces)
Major Architectural Feature:
Service Abstraction Layer (SAL)

Hard SAL (AD-SAL)


MD SAL
RESTCONF APIs

Base Network Service Functions

- Topology Mgr
- Stats Mgr
- Switch Mgr
- Fwding Rules Mgr
- Host Tracker
- ARP Handler

GUI

REST APIs

- oDMC
- AD-SAL

OpenFlow

1.0

VTN Coordinator

NETCONF

RESTCONF APIs

- LISP
- Base Network Service Functions
- Affinity
- VTN Mgr
- OVSDB
- CONF
- OF
- Base Network Functions
- BGP PCEP
- MD-SAL

LISP

REST

OpenFlow 1.0

SNMP

OVSDB

OpenFlow 1.0

1.3

NETCONF

BGP PCEP

OpenFlow Enabled Devices

Open vSwitches

Additional Virtual & Physical Devices

OF: OpenFlow
D4A: Defense for All
VTN: Virtual Tenant Network
oDMC: Open Dove Management console
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Protocol
SNMP: Simple Network Management Protocol
Who is OpenDaylight? (Members)

PLATINUM MEMBERS

GOLD MEMBERS

SILVER MEMBERS

HUAWEI
Who is OpenDaylight? (Really)

- Like any Open Source Project, OpenDaylight primarily consists of those who show up to do the work.
  - Currently commits from over 180 contributors from many different organizations (and unaffiliated individuals)

- Running around 100 commits per week
  - 30 Days: 369 commits, 50 contributors
  - 12 Months: 6415 commits, 181 contributors

- Strong integration and testing community
  - This stuff really matters

http://www.ohloh.net/p/opendaylight
The Hydrogen Simultaneous Release

- First release of OpenDaylight on February 3rd, 2014
  - Codename: Hydrogen
  - 15 different projects
  - Lots of integration and testing

- Several “editions” to group related functionality together
  - base, virtualization, service provider
  - virtualization edition provides OpenStack integration

- We all learned A LOT
Impressive List of Projects in $H_2$

- Controller
- VTN
- OpenDove
- Affinity Management Service
- LISP Mapping Service
- Yang Tools
- Defense4All
- BGP-LS/PCEP
- OpenFlow Protocol
- OpenFlow SB Plugin
- OVSDB
- SNMP4SDN
- DLUX
Hydrogen Release

Base Network Service Functions
- Topology Mgr
- Stats Mgr
- Switch Mgr
- FRM
- Host Tracker
- ARP Handler
- Affinity Service
- OpenStack Service
- Open vSwitches
- OpenFlow
- OpenFlow Enabled Devices
- OVSDB
- NETCONF
- SNMP
- BGP
- PCEP
- LISP
- Southbound Interfaces & Protocol Plugins
- Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Service Abstraction Layer (SAL)
(plug-in mgr., capability abstractions, flow programming, inventory, ...)

Network Applications Orchestration & Services
- VTN Coordinator
- D4A Protection
- OpenStack Neutron
- Controller Platform

OpenDaylight APIs (REST)

VTN: Virtual Tenant Network
oDMC: open Dove Management Console
D4A: Defense4All protection
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch Data Base Protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Communication Protocol
SNMP: Simple Network Management Protocol

Additional Virtual & Physical Devices
Base Edition

OpenDaylight APIs (REST)

Management GUI/CLI

Controller Platform

Network Applications Orchestration & Services

Southbound Interfaces & Protocol Plugins

Base Network Service Functions

- Topology Mgr
- Stats Mgr
- Switch Mgr
- FRM
- Host Tracker
- ARP Handler

Service Abstraction Layer (SAL)
(plug-in mgr., capability abstractions, flow programming, inventory, …)

OpenFlow
- 1.0
- 1.3

NETCONF

OVSDB

OpenFlow Enabled Devices

Open vSwitches

Additional Virtual & Physical Devices

Data Plane Elements
(Virtual Switches, Physical Device Interfaces)

VTN: Virtual Tenant Network
oDMC: open Dove Management Console
D4A: Defense4All protection
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch Data Base Protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Communication Protocol
SNMP: Simple Network Management Protocol
Virtualization Edition

Base Network Service Functions
- Topology Mgr
- Stats Mgr
- Switch Mgr
- FRM
- Host Tracker
- ARP Handler
- Affinity Service
- OpenStack Service

Service Abstraction Layer (SAL)
(Plug-in mgr., capability abstractions, flow programming, inventory, …)

OpenFlow
1.0 1.3

NETCONF
OVSDB

OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices

Controller Platform
- Network Applications Orchestration & Services
- Southbound Interfaces & Protocol Plugins
- Data Plane Elements (Virtual Switches, Physical Device Interfaces)

VTN: Virtual Tenant Network
oDMC: open Dove Management Console
D4A: Defense4All protection
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch Data Base Protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Communication Protocol
SNMP: Simple Network Management Protocol
OpenStack Integration

- OpenDaylight exposes a single common OpenStack Service Northbound
  - API exposed matches Neutron API precisely
  - multiple implementations of Neutron networks in OpenDaylight

- OpenDaylight OpenStack Neutron Plugin simply passes through
  - simplifies OpenStack plugin
  - pushes complexity to OpenDaylight
OpenStack Integration: Status

- **ML2 Driver available in Icehouse release!**
  - Supports VXLAN and GRE tunnel networks
  - devstack support merged upstream
    - *Run OpenDaylight as a top-level service in devstack!*
- **OpenStack Neutron API Service** available now in OpenDaylight
  - provides Neutron API handling for multiple implementations
- Initial ML2 plugin focused on core Neutron functionality
  - Still uses Neutron [DHCP, L3] agents
OpenStack Integration: Next Steps

• Updates planned for Helium and Juno:
  • VIF plugging changes for stability improvements
    • Notify from ODL to MechanismDriver once ODL has setup the port on the host
  • Security groups implemented using OpenFlow rules
  • L3 routing handled by OpenDaylight
    • Removes the need for the L3 agent
  • Additional refinements and bug fixes
Agenda

- What is OpenDaylight
- Some things I’ve learned
- A Few OpenDaylight Metrics
- Where OpenDaylight is Going
- SDN Grand Challenges
Key Learnings

- **Community building** is a core objective
  - In fact, innovation through collaboration is one of the most powerful features of open source development
- **Code** is the coin of the realm
- **Engineering systems** are as important as artifacts

**Putting this all Together →**
http://www.sdncentral.com/education/david-meyer-reflections-opendaylight-open-source-project-brocade/2014/03/
Trend: Engineering artifacts are no longer the source of sustainable advantage and/or innovation.

Perhaps surprisingly, the “hyper-scale” and open source communities have taught me that actual artifacts (in our case network applications as well as HW/SW) are ephemeral entities and that the only source of sustainable advantage/innovation consists of:

- Engineering Systems
- Culture
- People/Process

http://en.wikipedia.org/wiki/Aeroelasticity - Flutter
Factories vs. Babies

- “Most vendors **develop product like an overly anxious parents making a baby.** There is a lot preparation and planning and once the baby is “born” the product requires ongoing attention to reach maximum potential.”
- “By comparison, … has **organized itself as a product factory**. Each product is the result of a unified production line and the next product or feature is just a year or two away. Each product builds on the previous product.”
  - Even faster in open source software.
  - Networking hasn’t seen this yet.

http://etherealmind.com/difference-arista-competitors-factories-babies/
One way to think about open source development: Early on things are chaotic, there is a lot of stuff orbiting anything with enough gravity, there are epic collisions and everything is molten (e.g., like the surface of the earth during the LHB [0]). But if you wait a couple of billion years and let things evolve you can wind up with a beautiful blue planet or the Linux kernel or ...

[0] http://en.wikipedia.org/wiki/Late_Heavy_Bombardment
Transparency

- Transparency matters

- When there are disagreements in the community
  - Transparency makes everyone feel heard
  - Transparency makes sure the community does not fracture

- OpenDaylight is transparent to the extreme
  - All calls, mailing lists, wikis, etc. are open to the public
  - Even the technical steering committee calls
Agenda

- What is OpenDaylight
- Some things I’ve learned
- A Few OpenDaylight Metrics
- Where OpenDaylight is Going
- SDN Grand Challenges
OpenDaylight project creation

7 new project proposals pending

Slides courtesy of Anees Shaikh:
https://www.youtube.com/watch?v=fKC6WGcEjHE
http://events.linuxfoundation.org/sites/events/files/slides/OpenDaylight-Year1%20v4-ext.pdf
OpenDaylight code volume (ohloh.net)

Languages

- Total Lines: 1,548,552
- Number of Languages: 18
- Code Lines: 1,045,938
- Percent Code Lines: 67.5%
- Total Comment Lines: 322,675
- Percent Comment Lines: 20.8%
- Total Blank Lines: 179,939
- Percent Blank Lines: 11.6%

Code, Comments and Blank Lines

- Java: 461,768
- C++: 291,849
- C: 146,115
- Python: 33,539

Graph showing trend from 2013 to 2014.
In a Nutshell, OpenDaylight...

-- has had 4,759 commits made by 154 contributors representing 1,045,938 lines of code

-- is mostly written in Java with an average number of source code comments

-- has a young, but established codebase maintained by a very large development team with stable Y-O-Y commits

-- took an estimated 292 years of effort (COCOMO model) starting with its first commit in November, 2012 ending with its most recent commit 13 days ago
Membership — who wants to play

35

28

26

20

18

15

April 8 launch

June 3

June 5

June 15

October 3

January 16

February 4

www.opendaylight.org

www.opendaylight.org

Agenda

- What is OpenDaylight
- Some things I’ve learned
- A Few OpenDaylight Metrics
- Where OpenDaylight is Going
- SDN Grand Challenges
Current Projects

14 more project proposals in some state of preparation

- Controller
- Virtual Tenant Network (VTN)
- Open DOVE
- OpenFlow Plugin
- Affinity Metadata Service
- YANG Tools
- LISP Flow Mapping
- OVSDB
- OpenFlow Protocol Library
- BGP-LS/PCEP
- Defense4All
- SNMP4SDN
- Integration Group
- Dlux
- Group-based Policy
- OpenDaylight Toolkit
- PacketCable PCMM
- OpFlex Implementation
- Documentation
- Dynamic Resource Reservation
- Table Type Patterns (TTPs)
- SDNi
- OpenContrail

(Red are new since Hydrogen)

https://wiki.opendaylight.org/view/Project_Proposals:Main
Other Future Technical Work

- Core Infrastructure
  - Factoring apart the controller, e.g., MD-SAL, etc.
  - Data Persistence, DOM manipulation, etc.
  - Distributed Systems (Infinispan, Akka, …)
  - Performance, Scalability, Stability
  - Code Quality, Test Coverage,

- Python OpenDaylight Client

- We need more code that writes code
  - MD-SAL is an example
  - Fewer humans in the loop
  - More automation is more better
Non Technical Work

- Continue to build/refine our community
  - Increasing committer diversity across projects
    - Weekly status meetings
    - More transparency is more better
  - “Staffing”
    - Release engineering
    - Documentation

- Continue to refine our engineering systems
  - Thanks Andrew!
  - Fewer humans in the loop
  - SDN Simulation Platform
Agenda

- What is OpenDaylight
- Some things I’ve learned
- A Few OpenDaylight Metrics
- Where OpenDaylight is Going
- SDN Grand Challenges
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**
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 interop, 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.
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

https://www.youtube.com/watch?v=bcaBS6w_k_o
http://events.linuxfoundation.org/sites/events/files/slides/TTPs%20and%20NBIs%20for%20ods2014-final_0.pdf
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
Conclusions

- OpenDaylight is a rapidly growing open source community
  - Already shipping a large suite of tools to build SDN solutions
  - By most metrics, on-par with other successful open source

- Community, process and culture matters more than code
  - Focus on being able to adapt, fix, and ship the next thing

- SDN Grand Challenges
  - Centralized vs. Distributed
  - How to get there from here
  - Hardware Diversity
  - Application Composition