OPEN DAYLIGHT

OpenDaylight, OpenSource, and Why OSS is Important

ONS Accelerate Workshop

10 Feb 2015

http://www.opennetsummit.org/ons-accelerate-feb15.php

David Meyer

Chief Scientist and CTO, Brocade

Chair, OpenDaylight Board of Directors

Agenda

- What/Who is OpenDaylight?
- Why Open Source?
- Current OpenDaylight Release: Helium
- Open Source Lessons
- SDN Grand Challenges, and What Lies Beyond

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	Acceptance	Community
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.

Who is OpenDaylight?

PLATINUM MEMBERS

GOLD MEMBERS

SILVER MEMBERS

Who is OpenDaylight? (Really)

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

- Running around 150–200 commits per week
  - **30 Days:** ~400 commits, ~55 contributors
    - During releases this is >= 1000 commits and >= 100 committers
  - **12 Months:** ~10,000 commits, ~260 contributors

- Strong integration and testing community
  - This stuff *really* matters

Source: https://www.openhub.net/p/opendaylight

Agenda

- ~~What/Who is OpenDaylight?~~
- Why Open Source?
- Current OpenDaylight Release: Helium
- Open Source Lessons
- SDN Grand Challenges, and What Lies Beyond

Why Open Source?

- **Short version:** this is how modern infrastructure is built
  - “Undifferentiated Plumbing”
- **Longer version:**
  - Build more, better code faster via collaboration
  - Make better decisions with devs and users at the table
  - Spend more time on the code that matters
    - 80/20 rule: 80% of code is non-differentiating

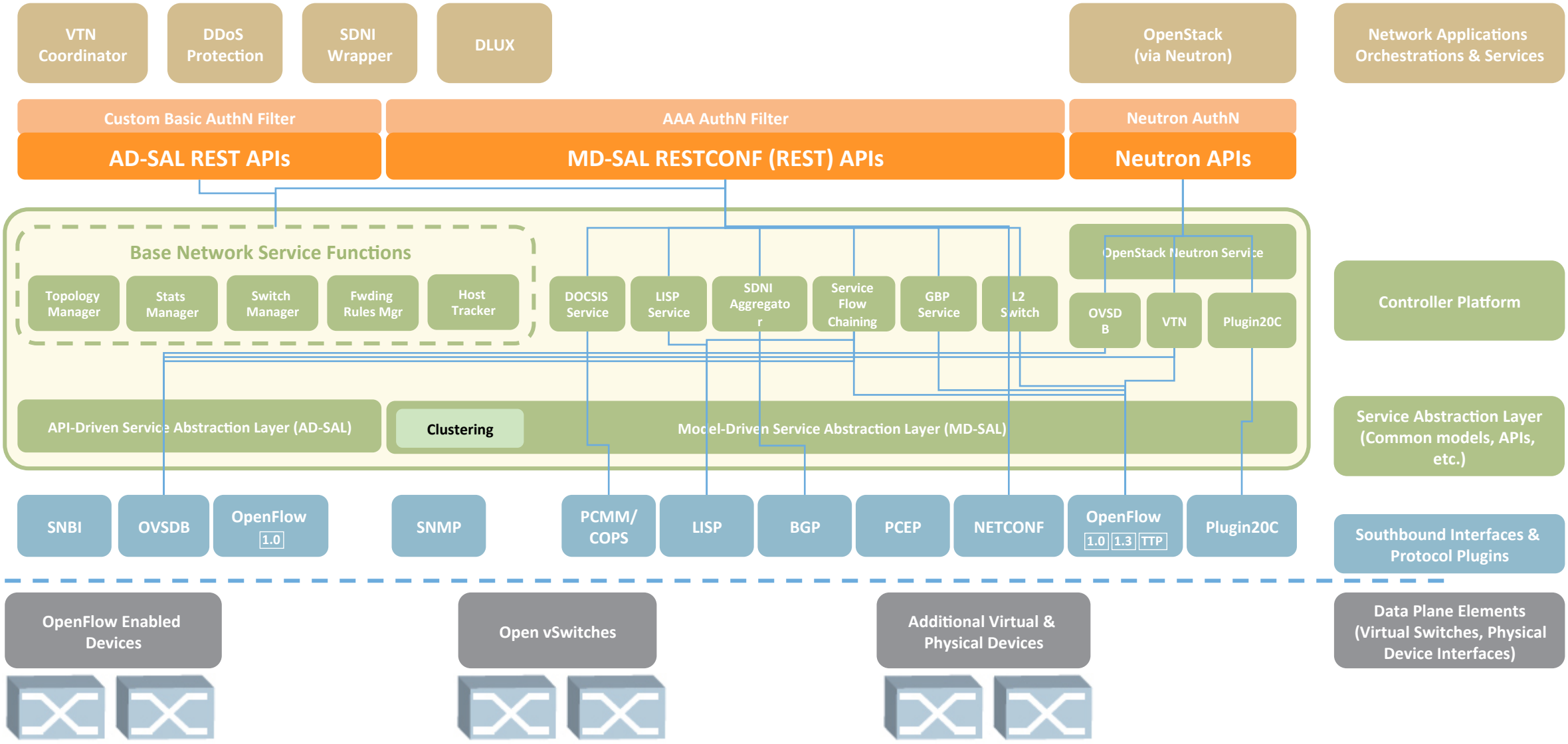
Agenda

- ~~What/Who is OpenDaylight?~~
- ~~Why Open Source?~~
- Current OpenDaylight Release: Helium
- Open Source Lessons
- SDN Grand Challenges, and What Lies Beyond

OPEN "HELIUM" DAYLIGHT

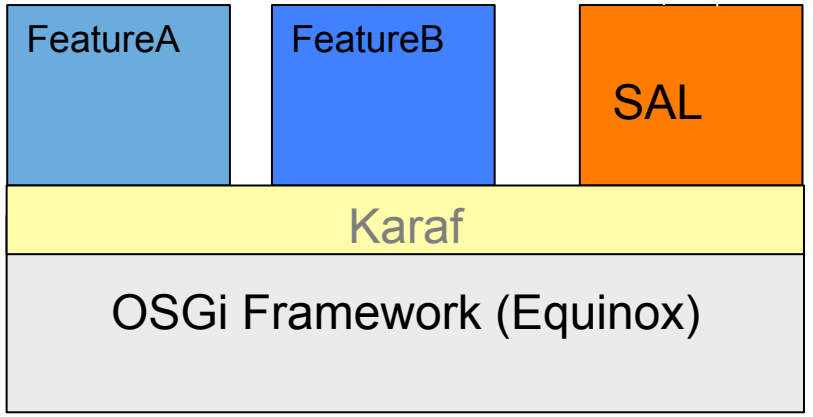
Legend

AAA: Authentication, Authorization & Accounting
AuthN: Authentication
BGP: Border Gateway Protocol
COPS: Common Open Policy Service
DLUX: OpenDaylight User Experience
DDoS: Distributed Denial Of Service
DOCSIS: Data Over Cable Service Interface Specification
FRM: Forwarding Rules Manager
GBP: Group Based Policy
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase Protocol
PCEP: Path Computation Element Communication Protocol
PCMM: Packet Cable MultiMedia
Plugin2OC: Plugin To OpenContrail
SDNI: SDN Interface (Cross-Controller Federation)
SFC: Service Function Chaining
SNBI: Secure Network Bootstrapping Infrastructure
SNMP: Simple Network Management Protocol
TTP: Table Type Patterns
VTN: Virtual Tenant Network



ODL Helium: Karaf

- Java chosen as an enterprise-grade, cross-platform compatible language
- Java Interfaces are used for event listening, specifications and forming patterns
- Maven – build system for Java
- OSGi:
  - Allows dynamically loading bundles
  - Allows registering dependencies and services exported
  - For exchanging information across bundles
- Karaf: Light-weight Runtime for loading modules/bundles
  - OSGi based. Primary distribution mechanism for Helium



ODL Helium: Karaf

```
$ wget
  http://nexus.opendaylight.org/content/groups/public/org/opendaylight/
  integration/distribution-karaf/0.2.0-Helium/distribution-karaf-0.2.0-
  Helium.zip
$ unzip distribution-karaf-0.2.0-Helium.zip
$ cd distribution-karaf-0.2.0-Helium
$ ./bin/karaf
opendaylight-user@root> feature:list  (get all apps available)
opendaylight-user@root> feature:install odl-dlux-core
opendaylight-user@root> feature:install odl-openflowplugin-all
opendaylight-user@root> feature:install odl-l2switch-all
opendaylight-user@root> bundle:list | grep Active
```

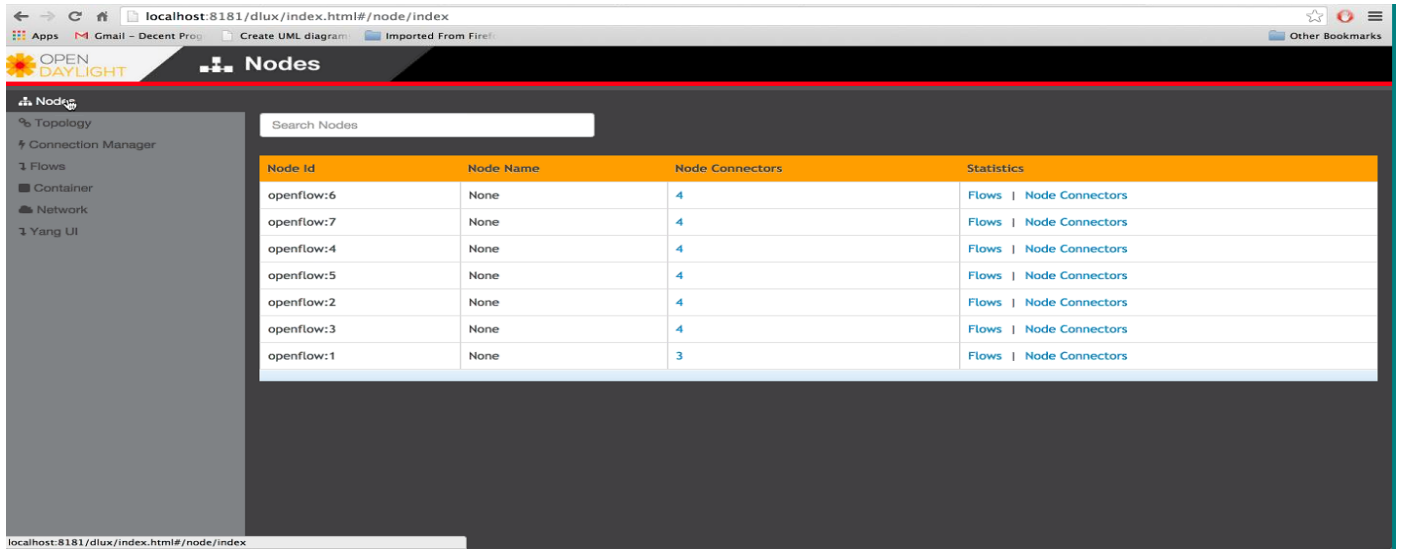
Now your controller is ready to connect to switches and handle incoming flows.

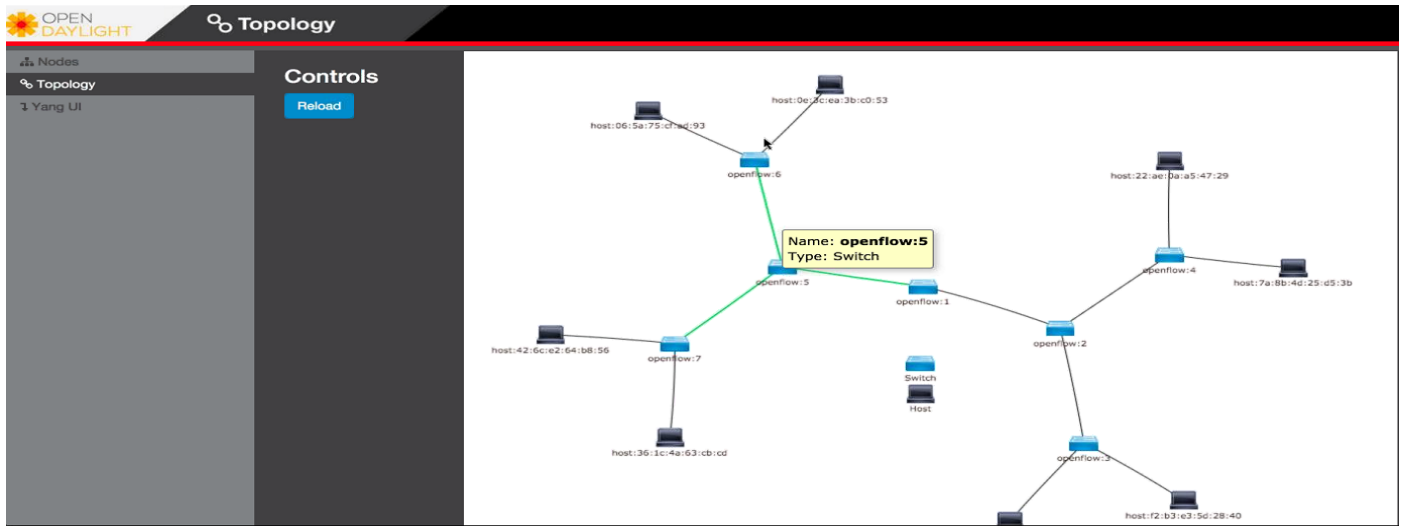
ODL Helium: Clustering

- The MD-SAL data store, notifications and RPCs now work in a cluster
  - Built using the RAFT consensus algorithm on top of Akka messaging
  - Tolerates f controller failures if you have 2f+1 controllers
  - Uses sharding for scale-out performance

- Lithium work items
  - Finer-grained, configurable sharding
  - Migrating plugins to take advantage of clustering and support failover
  - Provide clearer models for building clustered applications

ODL Helium: DLUX



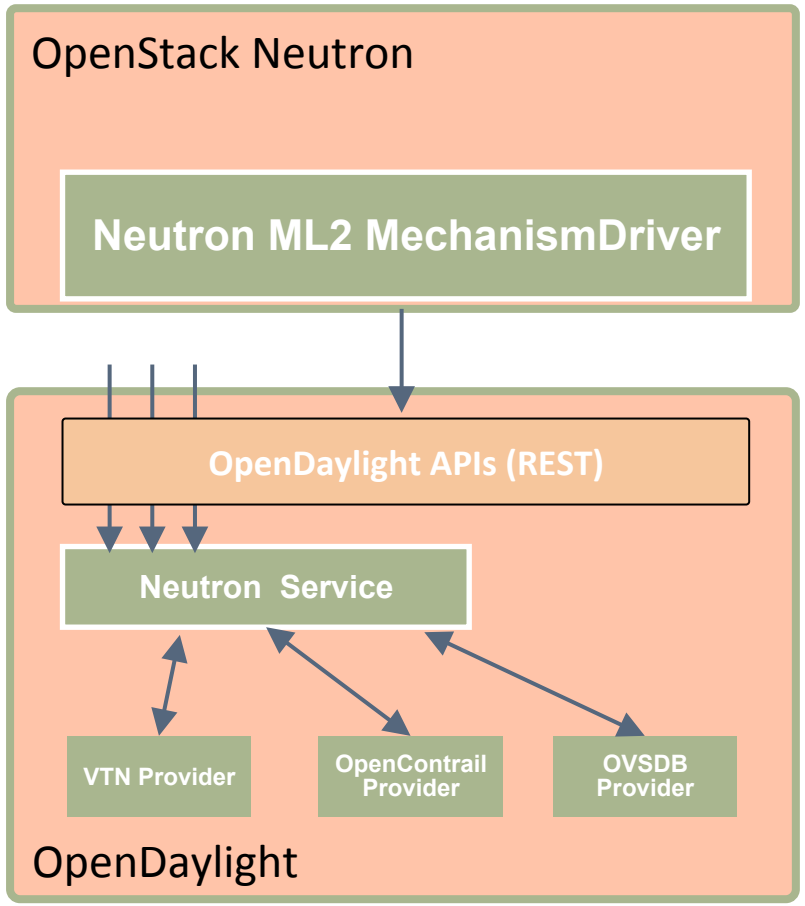


- Based on modern frameworks: node.js, AngularJS
- Completely decoupled from the core controller
  - Run it from any location
  - Modular, easy to extend

ODL Helium: Policy

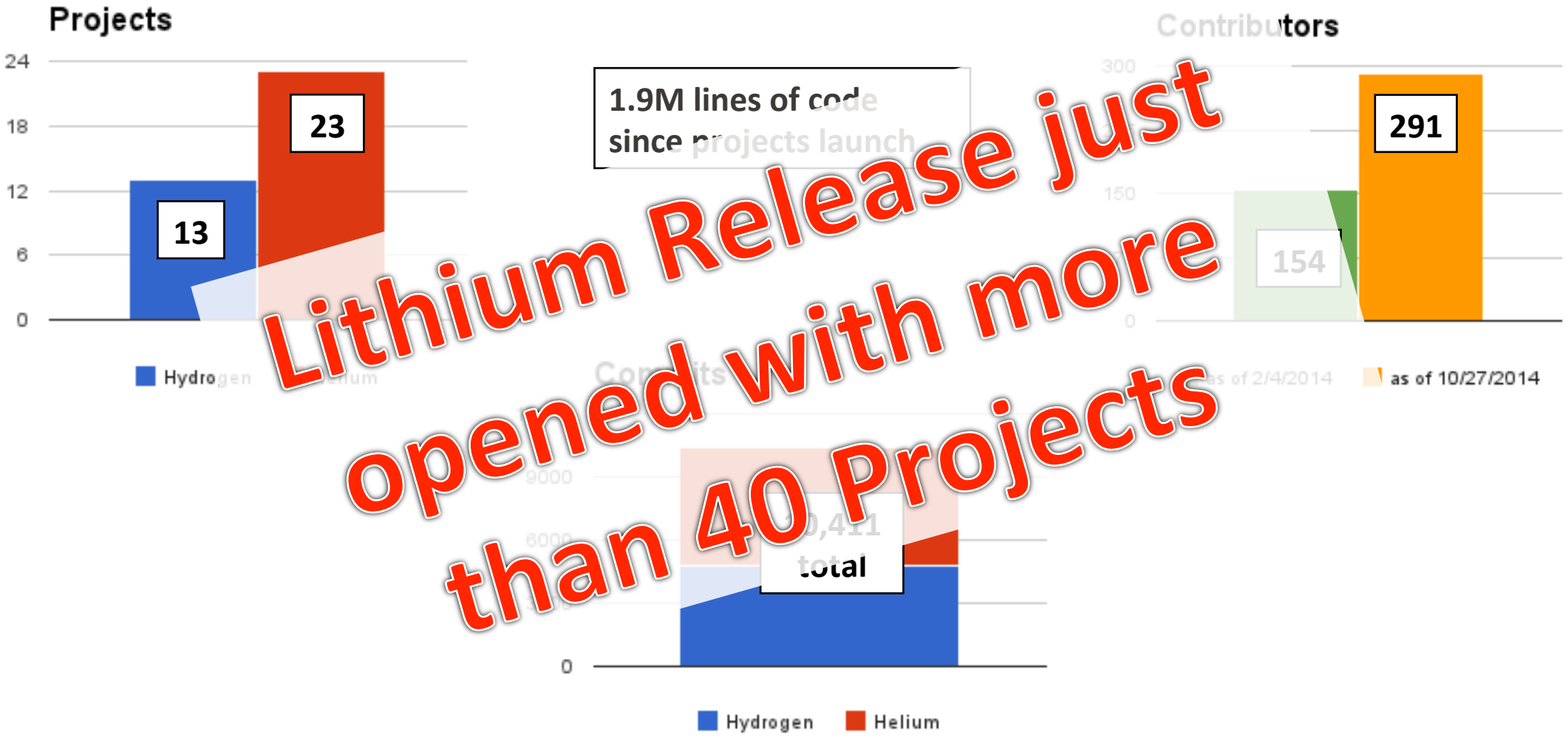
- Policy is everywhere at them moment
  - Group-based Policy, Congress, Intent, ACI, …
- At least three policy-oriented projects in ODL
  - Service Function Chaining
  - Group-based Policy
  - Network Intent Composition
- ODL is acting as a proving ground for policy approaches where engineers and users can play with different approaches

ODL Helium: OpenStack Integration



- OpenDaylight exposes a single common OpenStack Service Northbound
  - Matches Neutron API precisely
  - Multiple implementations of Neutron in OpenDaylight
- New features in Helium
  - Distributed L3 forwarding
  - OpenStack Security Groups
  - LBaaS implementation

Growth from Hydrogen to Helium



Lithium Release just opened with more than 40 Projects

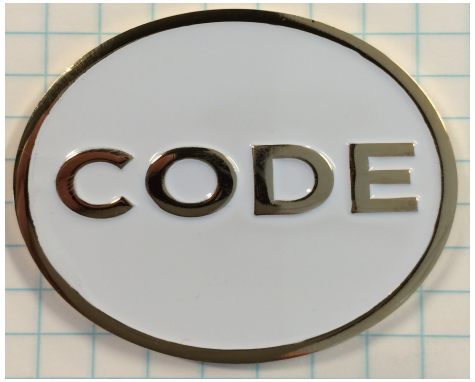
Adoption

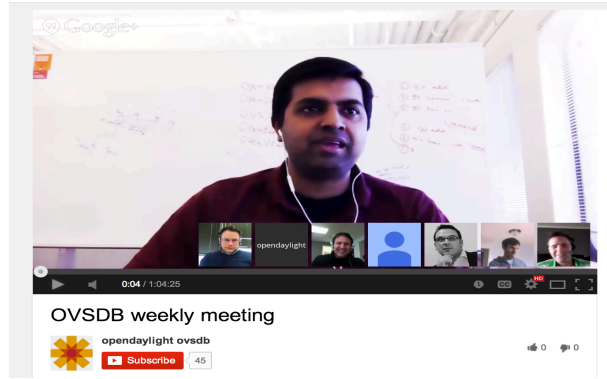

ADVA
Optical Networking
BROCADE
ciena
CISCO
CYAN
ERICSSON
Extreme
networks
IBM
INOCYBE
TECHNOLOGIES
midokura
ORACLE

Agenda

- ~~What/Who is OpenDaylight?~~
- ~~Why Open Source?~~
- ~~Current OpenDaylight Release: Helium~~
- Open Source Lessons
- SDN Grand Challenges and What Lies Beyond

Key Personal Learning:
Open Source is the Modern Way to Develop Non-Differentiated "Plumbing"

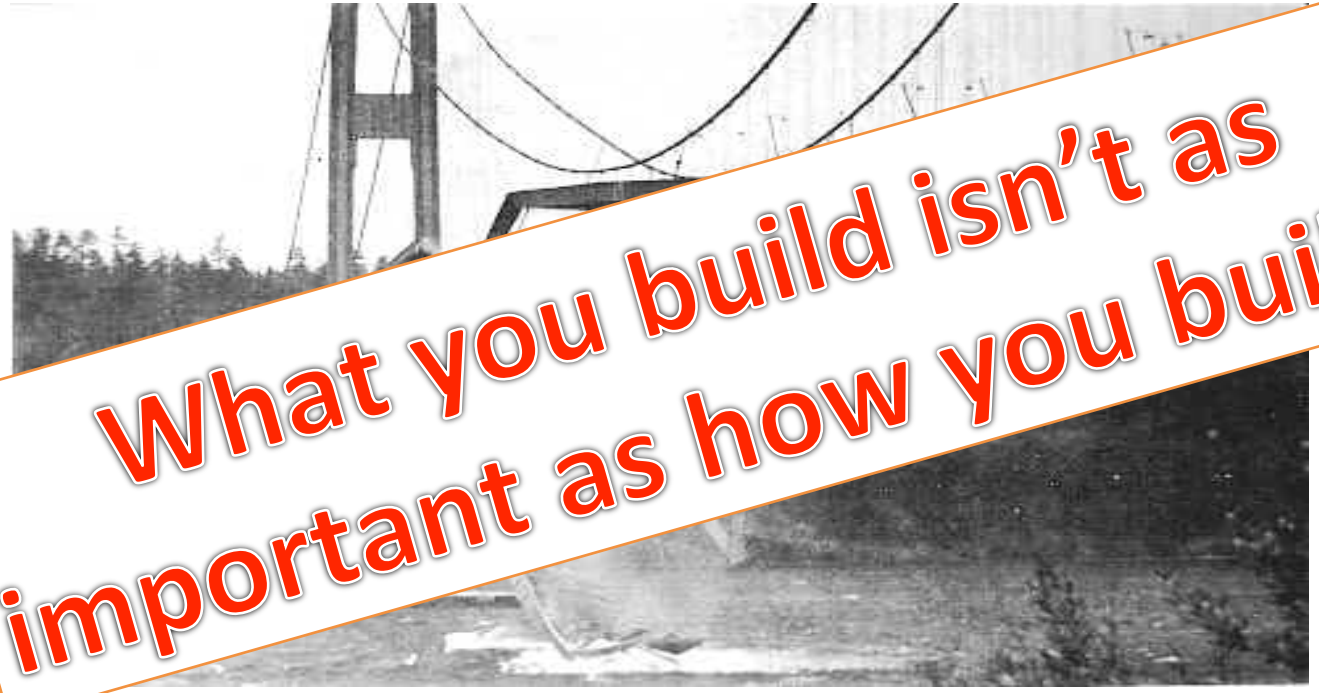




- ***Community building*** **is a core Open Source objective**
- ***Code*** **is the coin of the realm**
- ***Engineering systems*** **are as important as artifacts**

Putting this all together →

Implication: Engineering artifacts are *no longer* the source of sustainable advantage and/or innovation



What you build isn't as important as how you build it

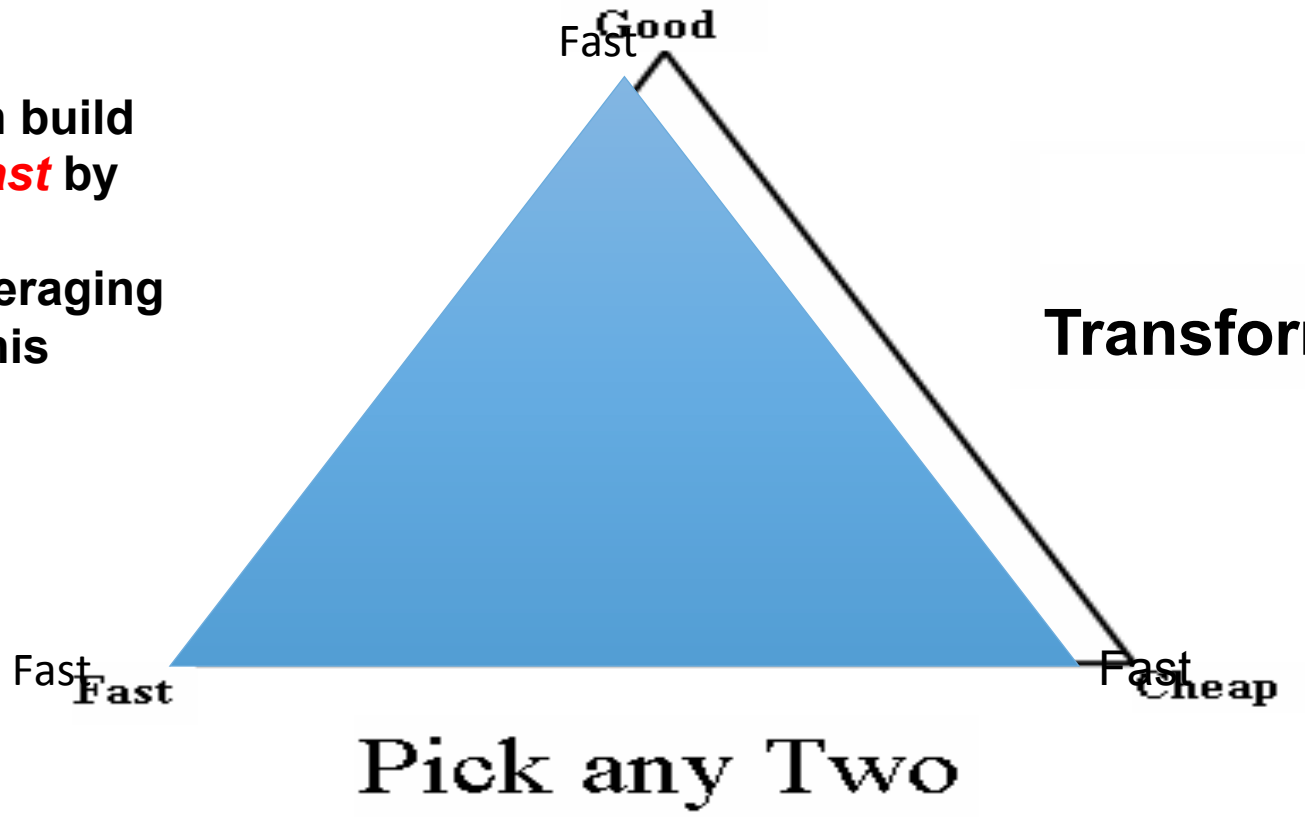
…(in our
…ations as well as
…are ephemeral entities and
that the only source of sustainable advantage/innovation consists of

- Engineering Systems
- Culture
- People/Process

http://www.sdncentral.com/education/david-meyer-reflections-opendaylight-open-source-project-brocade/2014/03/

Said Another Way:
Open Source has Transformed the Good-Cheap-Fast Development Cycle

Why? Because you can build *Good* or *Cheap* from *Fast* by using OS Development methodologies and leveraging the OS communities (this is a form of leveraged Investment)



Transformation →

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
  - Calls, mailing lists, wikis… are open to anyone
  - Even the technical steering committee calls

Agenda

- ~~What/Who is OpenDaylight?~~
- ~~Why Open Source?~~
- ~~Current OpenDaylight Release: Helium~~
- ~~Open Source Lessons~~
- SDN Grand Challenges and What Lies Beyond

SDN Grand Challenges

- Centralized vs. Distributed operation
  - RAFT distributed consensus algorithm in Helium
  - Continued work on clustering in Lithium and beyond
- Migration to SDN/Brownfield deployments
  - Currently support SNMP, BGP, LISP, NETCONF
  - Working on hybrid mode OF and others in Lithium
- Application Composition
  - Support for declarative, intent-based policy in Helium
  - Unified models for inventory and topology
  - Working on even better unified modeling in Helium
- Hardware Diversity
  - Initial support for Table Type Patterns in Helium
  - Device Driver Framework will provide adaptation in Lithium

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***

http://events.linuxfoundation.org/sites/events/files/slides/sdn-consistency-ods2014.pdf
https://www.youtube.com/watch?v=XQ-lnB3x30g

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
http://arxiv.org/pdf/1312.1719v1.pdf

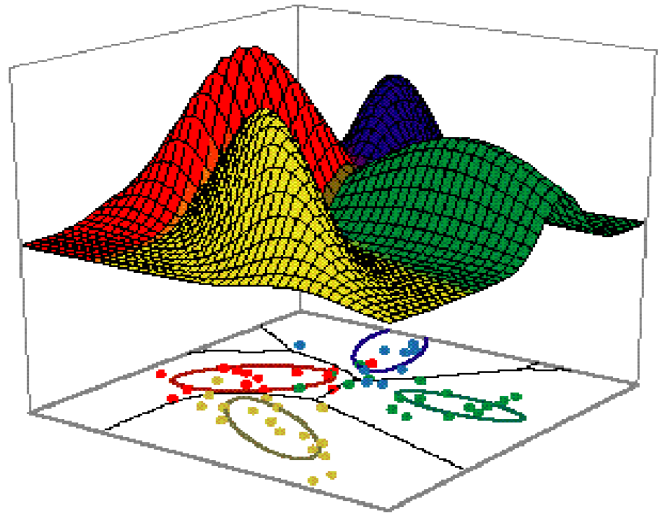
Application Composition

- 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

What Lies Beyond?

- Our goal was never to do the same thing
  - Only in a different way
- We want to build much smarter networks
  - But How?



- Software Defined Intelligence (SDI)
  - http://www.1-4-5.net/~dmm/talks/nfd8.pptx
  - http://techfieldday.com/appearance/brocade-presents-at-networking-field-day-8/
  - https://dmm613.wordpress.com/2014/09/17/software-defined-intelligence/

Q&A

Thanks!