OpenDaylight and the Rise of Open Source, Software Networking

Colin Dixon
Technical Steering Committee Chair, OpenDaylight
Senior Principal Engineer, Brocade

Some content from: David Meyer, Neela Jaques, and Kevin Woods
Outline

• Introduction...
  • ...to SDN
  • ...to OpenDaylight...
    • ...with a brief aside into YANG

• Open Source

• What’s in OpenDaylight

• Who’s using OpenDaylight

• Brocade’s SDN Controller
Traditional SDN (OpenFlow)
The separation of the control and data planes

- Modern switches
  - Control/data plane both on switch
  - Data plane: fast, reads tables
  - Control plane: slow, writes tables

- SDN
  - Decouple control/data planes
  - Data plane on the switch
  - Control plane elsewhere, e.g., an x86 server, can do fancier things
Modern, Inclusive SDN

Vendor A  Vendor B  Vendor C

gmt
control
gmt
control
gmt
control

Logically Centralized SDN Controller

Northbound API

Vendor A  Vendor B  Vendor C

mgmt
control

mgmt
control

mgmt
control

Industry Standard Control/Management Protocols

Standard Modeling Language
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.
OpenDaylight Releases

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

- **Helium** (second release)
  - October 2014
  - 25 projects, 2.1m lines of code

- **Lithium** (latest release)
  - June 2015
  - 40+ projects, 2.3m lines of code
Core Architecture

Model-Driven Service Abstraction Layer (MD-SAL)

- Notifications
- RPCs

Data

Controllers in a Cluster

App/Service

Plugin

RPCs

YANG Models

OpenDaylight
What is YANG?

• Data modeling language for NETCONF
  • RFC 6020

• Great, what is NETCONF?
  • Think of it as an SNMP replacement with nice features
  • YANG models \(\sim\) SNMP MIBs

• OK, fine, but what is YANG?
What is YANG?

• Three core abstractions
  • Data
  • RPCs (just data in and data out)
  • Notifications (just data out)

• So, it’s really all about the data
What does YANG data look like

- container ~= struct
- list ~= map/dictionary
- leaf ~= primitive types
- grouping ~= interface
- Others: typedef, pointers, constraints, etc.
Open Source
OpenDaylight Community

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

• Running around 250 commits per week over 12 months, trending up
  • 30 Days: ~625 commits, ~100 contributors (7/13/15–8/12/15; during a release)
    • Spikes to ~2x this near releases
  • 12 Months: ~13,250 commits, ~365 contributors (8/12/14–8/12/15)

• Strong integration and testing community
  • This stuff really matters

Source: https://www.openhub.net/p/opendaylight
Defining “Open” in Open Source

As about Projects

• Who can contribute?
• Who does contribute?
• How are decisions made? Who can comment? Who can vote?
• What license does it use?

Ask about Products

• Does it integrate with other solutions from other vendors?
• Does it have an API?
• Does it follow open standards?
• Is it based on open source components?
• Does it upstream to open source projects?

“Simply stated, OpenDaylight is as open as open gets.”

• [http://www.opendaylight.org/blogs/2014/03/degrees-open](http://www.opendaylight.org/blogs/2014/03/degrees-open)
Why Open Source?

• Avoid vendor lock-in

• Have a seat at the table

• Faster Innovation

• Interop & Integration
What’s in OpenDaylight
Kernel

- AAA
- YANG Tools
- OpenDaylight Controller
- MD-SAL
- NETCONF
- ODL Root Parent
Plugins

- BGP
- CAPWAP
- FaaS
- IoTDM
- LACP
- LISP
- OVSDB

- OpenFlow
  - Circuit switching extensions
  - OF-CONFIG
  - Table Type Patterns
- PCEP
- PacketCablePCMM
- SNMP

- SXP
- Secure Network Bootstrapping
- TCPMD5
- USC
- YANG PUBSUB
Services

- Armoury
- Centinel
- Controller Shield
- DIDM
- Messaging4Transport
- NeutronNorthbound
- NeXt
- ODL-SDNi App
- VPNService
- TSDR
- Topology Processing Framework
- Persistence
Applications

• ALTO
• Defense4All
• Group Based Policy (GBP)
• L2 Switch
• NEMO
• NetIDE
• Network Intent Composition

• OpenDaylight dlux
• OpenDaylight Virtual Tenant Network (VTN)
• Reservation
• Service Function Chaining
• SNMP4SDN
• Unimgr
Metaprojects

• Controller Core Functionality Tutorials
• Documentation
• Integration/distribution
• Integration/Packaging
• Integration/Test
• RelEng/Autorelease
• RelEng/Builder
What do people use it for?
Who’s using OpenDaylight and Why

Survey Respondents
- 31% Telcos/Service Providers
- 24% Research/Academia
- 20% Enterprises
- 10% Services/Consulting
- 9% Software/Hardware
- 6% Other
In Prod

- AT&T
- Telstra
- CERN/LHC
- Tencent

ODL at WebScale - Tencent

- Who: One of largest web-scale companies in the world
- What: DataCenter Interconnect Controller
- How: MD-SAL application, leveraging PCEP
- Reach: >500M users each for WeChat and QQ

“We request all our partners to be OpenDaylight compatible by end of this year”

-Marty Ma, Chief Architect

#odsummit
OpenDaylight with OpenStack

• Single OpenStack Neutron service proxy
  • Handles most of the bookkeeping

• Choose your implementation
  • Group-based Policy
  • LISP
  • OVSDB
  • VPN Service (only for VPNaaS)
  • VTN

• Check it out (see the links for instructions)

Network-wide Security Policy

• Historically, policy is mostly
  • Rigidly enforced by the physical topology, e.g., firewall at the gateway
  • Configured “dynamically” via box-by-box Access Control Lists (ACLs)

• New policy efforts are changing this
  • Network Function Virtualization (NFV) and Service Function Chaining (SFC)
  • Automatically generated ACLs based on network-wide policy

• OpenDaylight is a proving ground for at least 3 policy-oriented projects
  • Service Function Chaining, Group-Based Policy, and Network Intent Composition
Programmable EMS and/or NMS

• Huge number of southbound protocol drivers
  • OpenFlow, NETCONF, OVSDB, SNMP, BGP, PCEP, PCMM/COPS, etc.

• With a little bit of effort, you can write “shell scripts” for your network to either gather information or automate tasks

• Automate triggering activities based on network events, e.g., quarantine a host with L2 ACLs based on information from an IDS
Analysts See Momentum

“OpenDaylight is quickly evolving into something formidable with good potential for mainstream relevancy.”
– Andrew Lerner, Gartner

“OpenDaylight may be the third center of gravity”
– Andrew Lerner, Gartner

“OpenDaylight has become the Linux of network stacks: the foundation upon which both network vendors and users build the next generation of products and services.”
– Kurt Marko, Forbes

An open source approach to software-defined networking (SDN) moved several steps closer this week to becoming a de facto standard.
– Mike Vizard, IT Business Edge
How to get involved
Ways to get involved

• **IRC:** #.opendaylight on freenode: [http://webchat.freenode.net/](http://webchat.freenode.net/)
• **Mailing lists:** [http://lists.opendaylight.org/](http://lists.opendaylight.org/)
• **Wiki:** [http://wiki.opendaylight.org/](http://wiki.opendaylight.org/)
• **Documentation:** [https://www.opendaylight.org/downloads](https://www.opendaylight.org/downloads)
  • On github: [https://github.com/opendaylight/docs/](https://github.com/opendaylight/docs/)
• **Git/Gerrit:** [http://git.opendaylight.org/](http://git.opendaylight.org/)
  • Create an account: [https://identity.opendaylight.org/carbon/user-registration/user-registration.jsp](https://identity.opendaylight.org/carbon/user-registration/user-registration.jsp)
• **Projects:** [https://wiki.opendaylight.org/view/Project_list](https://wiki.opendaylight.org/view/Project_list)
  • Individual pages have links to meeting times, code, bugs, IRC channels, etc.
• **Meetings:** [https://wiki.opendaylight.org/view/Meetings](https://wiki.opendaylight.org/view/Meetings)
Brocade SDN Controller
• Brocade SDN (formerly Vyatta) Controller GA’d January 2015
  • OpenFlow 1.0, 1.3 and NETCONF interfaces
  • High-velocity release cycles: every 6 weeks

• Aggressive posture on pricing and packaging
  • Free for 5 devices or less with 60 days support
  • Base controller price = $100 per device (switch/router) per year, includes support

• Brocade DevNet and GitHub support communities launched

• store.brocade.com for Controller and future apps

• Several Tier1 telcos, CSPs and financial customers
• Strong channel interest – 1/3 downloads are VARs or SIs
Brocade SDN Controller – How, Not What

Build the foundation and frame upstream
- Continuous build from ODL
- Contribute enhancements
- No forking or proprietary extensions
- Channel user requests into ODL workstream

Collaborative Innovation With Users
- Joint and custom development
- App design review, certification, and/or integration
- Developer tools and repo access
- Developer community resources

Getting Started With SDN
- Architectural planning
- Stability and support for whole controller environment
  - Support model assumes mixed-vendor environment
  - Full documentation
- Specialized education for Operations and Developers
Brocade Topology Manager 1.0 (free App)
Public Announcement: Sept. 15, 2015

• First App using new ReactJS-based UX
• Display discovered topology
• Create a list of nodes
• Simple search for nodes

• Upgrade directly to Flow Manager
Brocade Flow Manager 1.0
Public Announcement: Sept. 15, 2015

• Install and Manage OpenFlows
• Install and Manage Host-to-Host Intent Paths
• Install and Manage Switch-to-Switch flows
• Uses new ReactJS-based UX
Brocade Flow Optimizer

1. Data Center Devices
   - Send sFlow samples to the Collector(s)
2. sFlow Collector(s)
   - Collect volumetric flow sample data
3. SDN App with policy-based UI and REST APIs
   - Analyze & manage volumetric flows
4. Brocade Vyatta Controller
   - Program OpenFlow rules in MLX and ICX
   - Future: VDX
Community Resources

- **Developer.brocade.com**
- **github.com/brocade**
  - Wiki: links to
  - Download sites
  - Documentation
  - Brocade DevNet
  - AppDev Tutorials
  - Useful ODL pages
  - SDN AppDev sites

**Ruby Bvc Introduction - Programming Brocade SDN Controller With Ruby**

- May 21, 2015
- Brocade Vyatta Controller Repo https://github.com/B
  - 18 commits
  - 2 branches

- **branch: master**
  - BVC / +

- **Update README.md**
  - **chuck-a-black** authored 20 days ago
    - docs
      - created docs folder
    - images
      - image for md-sal app creation
    - .gitignore
      - Add a .gitignore to ignore some files
    - README.md
      - Update README.md

**Docker Image for BVC 1.3.0**

- May 13, 2015

**Docker Image for Brocade SDN Controller and Path Explorer**

- May 11, 2015

**vRouter 5600 Install**

- Apr 28, 2015

**Docker Image for Brocade SDN Controller**

- Apr 27, 2015

**Install Script for Brocade SDN Controller**

- Apr 20, 2015
Guiding Principles: Open, Trusted, Extensible

Community is the force multiplier

Value Increase Over Time

- Simple on-ramp to SDN
- Collaborative innovation

Current State

Efficiency, Automation, Service Agility

Self-Service Innovation

Leverage Community Contributions

Core Tech: Open, Secure, Reliable, Easy to Deploy

Flexibility to Experiment (technical, legal)

Help as Needed from Supportive Community

Trusted Support, Training

• Simple on-ramp to SDN
• Collaborative innovation
Open Research Questions
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

https://www.youtube.com/watch?v=bcaB56w_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