The State of (Open Source) SDN and Programming Language Opportunities

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

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

• What I know a lot about
  • Who I am and why I’m here
  • A brief history of SDN
  • Problems that I have

• What I know a lot less about
  • Ideas for how to solve them in “nice” ways
  • …instead of my normal systemsy hacks
Colin Dixon

- PhD Univ. of Washington 2011
- IBM Research Austin 2012–14
  - Started with ODL in late 2012
- OpenDaylight 2012-present
  - Committer-at-Large (9/2014)
  - TSC Chair (10/2014)
- Brocade 2014-present

- Grad Student
- Research Lab Academic
  - Working on products
  - Working in Open Source
- Industry
  - Working in Open Source
  - Shipping products
A brief history of SDN

- **Original Papers**
  - SANE/Ethane (2006/2007)
  - OpenFlow (2008)
  - Onix (2010)

- **(Open Source) Software**
  - Open vSwitch (2009)
  - NOX/POX (2009/2011)
  - Beacon (2010)
  - Trema (2011)
  - Floodlight (2011/2012)
  - Ryu (2011/2012)
  - OpenDaylight (2013)
  - ONOS (2014)

This is just Controllers
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

Logically Centralized SDN Controller

NETCONF/YANG

Northbound API

Industry Standard Control/Management Protocols

Standard Modeling Language
Acquisitions (very incomplete)

• 2012: Nicira => VMware (1.26B)
• 2012: Contrail => Juniper (176M)
• 2013: Insieme => Cisco (863M)
• 2014: Tail-F => Cisco (175M)
• 2015: Cyan => Ciena (400M)
Deployed in Production Today

- AT&T’s Domain 2.0 initiative
  - Broad attempt to move to SDN and NFV in a major way
- BWoD runs us Brocade’s OpenDaylight-based controller

- Also see:
  - Google, Microsoft, VMware, etc.
Summary

• SDN has gone from wild, crazy academic idea to reality
  • Tons of attention from academia and industry
  • Sustained ~1B/year in acquisitions

• Along the way it’s changed a lot
  • Not just about separation of the control plane from the data plane
  • Not just OpenFlow

• I think it’s the rise of open source software networking
Open Source SDN Projects of Note

- Akanda
- CloudRouter
- MidoNet
- ONIE
- ONOS
- OVN
- Open Compute
- Open Network Linux
- OpenContrail
- Prescriptive Topology Manager
- Quagga
- SocketPlane
- Weave
- ...and many, many more
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**
  - October 2014
  - 25 projects, 2.1m lines of code
- **Lithium** (most recent release)
  - June 2015
  - 40 projects, 2.3m lines of code
Model-Driven Service Abstraction Layer (MD-SAL)

App/Service

App/Service

Core Architecture

Controllers in a Cluster

Plugin

Plugin

Data

Notifications

RPCs

YANG Models
What does YANG data look like

- container ~= struct
- list ~= map/dictionary
- leaf ~= primitive types
- grouping ~= interface
- Others: typedef, pointers, constraints, etc.
Why Java?

• Obvious answer: it’s what we started in, so...

• Slightly deeper:
  • ~40 companies looking to hire ~10 developers each at some cost
  • Java developers are (a) common, (b) reasonably priced, (c) somewhat safe
  • CUDA/OpenCL : GPUs :: OpenDaylight : Java programmers

• Would we change to something else?
  • Not everywhere. In some places, maybe.
Programming Language Opportunities

Things that are currently causing me problems.
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 provides 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.
How to get there from here

• Flowlog’s Exodus, Header Space Analysis, etc.
  • Translate legacy config to SDN programs with correct semantics

• Works pretty well when everything is “SDN-enabled”
  • In practice, connected to legacy network devices
  • Also, middleboxes
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

https://www.youtube.com/watch?v=XQ-InB3x30g
Centralized vs. Distributed
(Consistency, Clustering and Federation)

• What properties can I provide when mixing consistency levels?
  • the topology is stored in an eventually consistent manner
  • the actions I take based on it should not diverge

• Is Irmin one answer?
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 (P4)
  • TTPs from the ONF’s FAWG

https://www.youtube.com/watch?v=bca856w_k_o
http://events.linuxfoundation.org/sites/events/files/slides/TTPs%20and%20NBIs%20for%20ods2014-final_0.pdf
Table Type Patterns

- Really just a simple way to express fixed OF 1.3 pipelines
  - DAG of tables
  - Legal jumps between tables
  - Legal matches in each table
  - Legal actions/instructions in each table

- Hope is to be able to map SDN services/apps to these in a sane way
  - Compiler? What’s the source language?
Are TTPs also the source language?

L2 Forwarding

DMAC => port

L2/L3 Fwding/Routing

DMAC => port

IP => nextHop

nextHop => port

Broadcom OF-DPA
10-table TTP

Netronome
Corsa
Marvell
Soon: HP, Dell, Intel...
Why is this hard?

• Some actions are impossible on a given switch
  • Can “fake” it in the controller
  • Better: use a fast programmable software switch
    • Could even be an end-host vSwitch, host stack, NIC
Application Composition

• How can we let multiple SDN apps share the network?
  • PC OSes partition and allocate resources
  • It’s value comes from the fact that it spans everything
  • You can in some cases, e.g., by address space (FlowVisor)

We’re systems people, we can’t help it!

• 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’s hope that conflicts can be managed sanely

Better Idea: DSLs?
Have: Neutron, NIC, GBP
Coming to ODL: NEMO, Maple, Pyreic
Application Composition

• The reality in OpenDaylight
  • Some apps written to h/w
  • Other apps written to DSLs
    • Many different DSLs

• Can I combine different DSLs?
• What about detecting conflicts with apps with no properties?
Call to Action
Open source => real impact quickly

• If you have ideas, they can be real pretty quickly
  • Switches: OF-DPA + Open Network Linux on Broadcom Hardware
  • Controllers: OpenDaylight, ONOS, etc.
  • Orchestration: OpenStack, etc.

• You don’t have to play with simulators
  • Build the real thing in real code that real people can pick up and use
OpenDaylight in particular

• Designed to be customized and productized
  • Ridiculously modular
  • Nobody will tell you no (at least to producing a new project)

• We could really use help, guidance and people.
Backup Slides
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
OpenStack Neutron Integration

- OpenDaylight has a common Neutron “northbound” provider
  - 3 implementations in Helium
  - 4+ implementations in Lithium

- Supports L2 network virtualization and
  - Distributed L3 forwarding
  - Security Groups
  - {LB,VPN}aaS
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
How can I get it?

- OpenDaylight: [http://www.opendaylight.org/software/downloads](http://www.opendaylight.org/software/downloads)

- Also commercialized, supported versions from
  - Brocade, Ciena, Cisco, Inocybe, and others
  - Understand the difference between “uses” and “based on” OpenDaylight
  - Policy on “upstreaming” changes and compatibility with other products

- The Brocade Vyatta Controller is based on unmodified OpenDaylight and upstreams all changes to OpenDaylight
  - Get it here: [https://tinyurl.com/BrcdVytaCntrlr](https://tinyurl.com/BrcdVytaCntrlr)
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