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

• State of networking and SDN
• Overview of open source (networking)
• Introduction to OpenDaylight
• Who’s using OpenDaylight and for what
• Open Research Questions
Networks have not adapted to demands

• Last 20 years → radically shifting network demands
  • massively increased scale (# of endpoints, switches, bytes, flows, etc.)
  • static endpoints (weeks–months) → dynamic endpoints (hours–days)
  • mostly north-south traffic → mostly east-west traffic

• By contrast, networks haven’t changed much
  • Link speeds have gone up, but...
  • Still largely manage networks device-by-device via the CLI
  • If you’re lucky, orchestration at the granularity of a few devices
Things need to change

• Device-by-device → Network-wide
• Open Standards → Open Standards + Open Source
• Proprietary Software → Open Source
• Networking, Storage, Compute → Converged IT
• Hardware → Software

• To a large extent, **this is the rise of open source networking**
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
**Disaggregation & Open Source Software**

- **Modern, Inclusive SDN**
- **Vendor A**
- **Vendor B**
- **Vendor C**

- **Logically Centralized SDN Controller**
- **Northbound API**
- **Industry Standard Control/Management Protocols**
- **Standard Modeling Language**

**Vendor A**
- **mgmt**
- **control**

**Vendor B**
- **mgmt**
- **control**

**Vendor C**
- **mgmt**
- **control**

**Device-by-device operation**
- Proprietary, vendor-specific vertical stacks for control, management and orchestration
  - Limited innovation in individual silos

**Network-wide operation**
- Open control, management and orchestration using open control protocols/modeling langs
  - Independent innovation at each layer of the stack
Open Source
Open Source Projects of Note

• **OpenStack**: IT-wide orch. (Neutron for networks)  
  
  [orchestration]

• **OpenDaylight**: industry-wide SDN controller  
  
  [control/mgmt plane]

• **Open vSwitch**: programmable s/w in the Linux kernel  
  
  [data plane]

• **Many, many others**: Open Network Linux, ONOS, CloudRouter, Quagga, OVN, ONIE, Open Compute, Prescriptive Topology Manager, SocketPlane, Weave, Akanda, MidoNet, OpenContrail

http://www.jedelman.com/home/open-source-networking
Why Open Source?

• Avoid vendor lock-in
• Have a seat at the table
• Faster Innovation
• Interop & Interoperability

My Thesis: This is really more important than the SDN technology itself.

You’ll not say cost

build it  open source  buy it
Open Source vs. Open Standards

Open Standards
• define interfaces well
  • in human-readable documents
• define behavior with some ambiguity
• usually move slowly
• leave interoperability testing to others, e.g., users, integrators
• sometimes provide open source implementations

Open Source
• define interfaces well
  • in code
• define behavior in code so it can be tested and understood
• move and adapt quickly
• can do interoperability testing as part of development
• often implement open standards
Open Source Opportunities

• SDN in production is now using the same tools that researchers and academics can use
  • Same is Linux and other examples
  • Potential for significant real-world impact

• Also lots of challenges
  • Real code means real complexity
  • Navigating getting code accepted “upstream” is hard
  • Sustained contributions work better, but are higher cost
OpenDaylight
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.
Core Architecture

Model-Driven Service Abstraction Layer (MD-SAL)

Controllers in a Cluster

App/Service

Plugin

Plugin

Data

Notifications

RPCs

YANG Models
OpenDaylight Community
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
What do people use it for?
In Production

- 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
Network Virtualization

• 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)

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
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)
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**

https://www.youtube.com/watch?v=XQ-lnB3x30g
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 while not burdening developers with per-device programming is hard

- Some Attempts
  - P4: Programming Protocol-Independent Packet Processors
  - TTPs from the ONF’s FAWG
  - Protocol Oblivious Forwarding (Huawei)

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