OpenDaylight as a Platform for Network Programmability

FOSDEM, 3 February 2018

Charles Eckel, Cisco DevNet
eckelcu@cisco.com
Agenda

• What is SDN
• What is OpenDaylight
• Network programmability
• Installation
• Example use case (VPP)
• Conclusions
What is SDN
Software Defined Networking (SDN)

• Control & Data Planes separation?
  • OpenFlow?
  • Logically centralized control Plane?
  • White label switches?

• This a valid & useful SDN use case, but...

• SDN can be defined more broadly:
  • Network is a source of vast amount of data...
  • ..that can be utilized by variety of SDN applications

• True power of SDN is network programmability
SDN - A Broader Definition

Application Developer Environment

- Management and Orchestration
- Analysis and Monitoring, Performance and Security
- Network Services
- Control Plane
- Forwarding Plane
- Network Elements and Abstraction
- Transport

Harvest Network Intelligence

Program for Optimized Experience

Generic feedback/control/policy loop between apps and the network
What Do We Need from an SDN Controller?

• A platform for deploying SDN applications

• Provide an SDN application development environment
  • Developer-friendly APIs to network elements (REST/JSON, pub/sub, etc.)
  • Network-level abstraction through topologies
  • Protocol independence for network-facing applications
What is OpenDaylight
The OpenDaylight Community

• Founded in February 2013
• Run by the Linux Foundation
• Eclipse Public License
• 15 founding companies provided software and developers
• 600+ contributors
• 2.5M+ lines of code
• Mostly Java

• First release “Hydrogen”
  • February 2014
• Release frequency
  • Roughly every 6 months
• Current release - “Nitrogen”
  • 7th release, Sept 26, 2017
  • SR1 released Nov 26, 2017
• Next release is Oxygen
  • March 2018
Software Architecture

- Java - enterprise-grade, cross-platform compatible language
- Java Interfaces - for event listening, specifications and forming patterns
- Maven – build system
- Karaf – based on OSGi, provides:
  - dynamic loading of bundles
  - registering dependencies and services exported
  - exchanging information across bundles
Network programmability
Why Network Programmability Matters

Network Expenses

- CAPEX: 33%
- OPEX: 67%

Source: Forrester

Deployment Speed

- Computing: 0 seconds
- Networking: 1000 seconds

Source: Open Compute Project
The Need for Something Better

- SNMP had failed
  - For configuration, that is
  - Extensive use in fault handling and monitoring
- CLI scripting
  - “Market share” 70%+

RFC 3535

Abstract

This document provides an overview of a workshop held by the Internet Architecture Board (IAB) on Network Management. The workshop was hosted by CNRI in Reston, VA, USA from June 4 thru June 6, 2002. The goal of the workshop was to continue the important dialog started between network operators and protocol developers, and to guide the IETFs focus on future work regarding network management.
Best Practices Coming Together

- CLI Best Practices
- SNMP Experience
- Operations Requirements
- NETCONF, RESTCONF and YANG
YANG
YANG
Data Modeling Language for Networking

• Modeling language, defined in RFC 6020
• Represents operational state, configuration, transactions, and notifications
• Defines semantics
  • Constraints (i.e. “MUSTs”)
  • Reusable structures
  • Built-in and derived types

In Summary:
YANG is a full, formal contract language with rich syntax and semantics for network data
Model Structure

- Data structured as a tree
- Main node types:
  - Container
  - List
  - Leaf List
  - Leaf

Node without a value

Node with a value
YANG Model Example

- Screenshot from ietf-interfaces.yang
- Container 'interfaces' with list of 'interface' items
- List items (leafs) have a 'name' which is also the key for the list
Tools to work with YANG Models

- **pyang** - An extensible YANG validator and converter in python
  - Source Code - [https://github.com/mbj4668/pyang](https://github.com/mbj4668/pyang)
  - Python Package - [https://pypi.python.org/pypi/pyang](https://pypi.python.org/pypi/pyang)

- **YANG Explorer** - YANG Browser / RPC Builder
  - [https://github.com/CiscoDevNet/yang-explorer](https://github.com/CiscoDevNet/yang-explorer)

- **OpenDaylight YANG Tools** – Tools supporting NETCONF and YANG, code generation from YANG models
  - [https://wiki.opendaylight.org/view/YANG_Tools:Main](https://wiki.opendaylight.org/view/YANG_Tools:Main)
Display a YANG Module

```
$ pyang -f tree <yang-file>

$ pyang -f tree <odl-dir>/cache/schema/ietf-interfaces\@2014-05-08.yang
module: ietf-interfaces
  +--rw interfaces
    |  +--rw interface* [name]
    |     +--rw name        string
    |     +--rw description? string
    |     +--rw type         identityref
    |     +--rw enabled?     boolean
    |     +--rw link-up-down-trap-enable? enumeration {if-mib}?
  +--ro interfaces-state
    +--ro interface* [name]
      +--ro name        string
      +--ro type        identityref
      +--ro admin-status enumeration {if-mib}?
      +--ro oper-status enumeration

[...]
```
pyang – jsTree Output

$ pyang -f jstree -o <output-file> -p <path to models> <model.yang>

$ pyang -f jstree -o ietf/interfaces.html -p ./cache/schema ./cache//schema/ietf/interfaces\@2014-05-08.yang
Building a Plugin/Application

1. **Generate APIs**
   - YANG Model
   - YANG Tools
   - Generated API Definition
     - Maven Build Tools
     - Plugin source code

2. **Create API Bundle**
   - “API” OSGI Bundle
     - Maven Build Tools

3. **Create Plugin Bundle**
   - “Plugin” OSGI Bundle
     - Plugin source code

4. **Deploy**
   - “API” OSGI Bundle
   - “Plugin” OSGI Bundle

OpenDaylight as a Platform for Network Programmability
© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public
NETCONF
NETCONF

IETF network management protocol

- Distinguishes between configuration and operational/state data
- Multiple configuration datastores (candidate, running, startup)
- Configuration change validation and transactions
- Selective data retrieval via filtering
- Streaming and playback of event notifications

In Summary:

NETCONF provides fundamental programming features for convenient and robust automation of network services
NETCONF Sessions

- NETCONF is connection-oriented
  - SSH, TLS as underlying transport
  - XML for payload

- NETCONF client establishes session with server

- Session establishment: `<hello>` exchange
  - Announce capabilities, modules, features

- Session termination
  - `<close-session>`, `<kill-session>`
NETCONF Commands

- get : to retrieve operational data
- get-config : to retrieve configuration data
- edit-config : to edit a device configuration
- copy-config : to copy a configuration to another data store (e.g. non-volatile memory)
- delete-config : to delete a configuration in a data store
RESTCONF
RESTCONF
Restful API for YANG data models

- IETF RFC 8040
- Configuration data and state data exposed as resources
- How to access the data using REST verbs (GET / PUT / POST/ …)
- How to construct URIs to access the data
- HTTP instead of SSH for transport
- JSON in addition to XML for data encoding

In Summary: RESTCONF provides lighter-weight interface to network datastores leveraging well known combination of REST and JSON
RESTCONF URI & JSON Example

```
pyang -f tree ietf-interfaces.yang

module: ietf-interfaces
  +--rw interfaces
     |  +--rw interface* [name]
     |     |  +--rw name string
     |     |  +--rw description?
     .
     .
     .

http://172.16.126.250:8008/api/running/interfaces

{ "ietf-interfaces:interfaces": {
  "interface": [
    {
      "name": "GigabitEthernet3",
      "description": "To CE-1"
    }
  ]
}}
```
High Level Manageability Architecture

Application

ANY (C, Java, Python)

- NETCONF client

ANY (Java, Python, Perl, PHP)

- RESTCONF client

Transport

- YANG-based XML
- SSH / TLS

- YANG-based XML/JSON
- HTTPS

Network Device

Manageability Infra

- BGP
- QoS
- VXLAN

Manageability

- Config DB

- Config DB

- Config DB

- Config DB

ANY (Java, Python, Perl, PHP)

- RESTCONF server

ANY (C, Java, Python)

- NETCONF server

OpenDaylight as a Platform for Network Programmability
© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public
Mounting YANG Datastores
OpenDaylight NETCONF Node “Discovery”

• Nodes added by POSTing to config:modules
• OpenDaylight connects to each node
• OpenDaylight learns capabilities (YANG modules) and stores to cache
  • Cache at ~/cache/schema. Filenames of form yang-model@2016-07-12.yang.
Installation
## Distributions

[https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation](https://www.opendaylight.org/technical-community/getting-started-for-developers/downloads-and-documentation)

### Downloads

<table>
<thead>
<tr>
<th>Release</th>
<th>Release date</th>
<th>Downloads</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carbon SR2</td>
<td>October 16, 2017</td>
<td>• Pre-Built Tar&lt;br&gt;• Pre-Built Zip&lt;br&gt;• NeXt UI&lt;br&gt;• Virtual Tenant Network (VTN) Coordinator</td>
<td>• Getting Started Guide&lt;br&gt;• Developers Guide&lt;br&gt;• User Guide&lt;br&gt;• Installation Guide&lt;br&gt;• Using OpenDaylight with OpenStack&lt;br&gt;• Release Notes</td>
</tr>
<tr>
<td>Nitrogen SR1</td>
<td>November 26, 2017</td>
<td>• Pre-Built Tar&lt;br&gt;• Pre-Built Zip&lt;br&gt;• Virtual Tenant Network (VTN) Coordinator&lt;br&gt;• OpFlex</td>
<td>• Getting Started Guide&lt;br&gt;• Developers Guide&lt;br&gt;• User Guide&lt;br&gt;• Installation Guide&lt;br&gt;• Using OpenDaylight with OpenStack&lt;br&gt;• Release Notes</td>
</tr>
</tbody>
</table>

*(Current Release)*
$ unzip karaf-0.7.1.zip
Archive: karaf-0.7.1.zip
        creating: karaf-0.7.1/system/ ...

$ cd karaf-0.7.1
$ ./bin/karaf
karaf: Enabling Java debug options: -Xdebug -Xnoagent -Djava.compiler=NONE
      -Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=5005
Listening for transport dt_socket at address: 5005
Apache Karaf starting up. Press Enter to open the shell now...
100% [=========================================================================]

Karaf started in 0s. Bundle stats: 10 active, 10 total

OpenDaylight

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or type 'system:shutdown' or 'logout' to shutdown OpenDaylight.

opendaylight-user@root>
Install Features using Karaf

- OpenDaylight distro comes without any features enabled by default
- All features are available for you to install
  - `feature:list`
  - `feature:list -i`
  - `feature:list -r`
  - `feature:install <feature>`
  - `feature:install <feature-1> <feature-2> … <feature-n>`
  - `feature:uninstall <feature>`

list all features available
list all features installed
list all features required
install the `<feature>` feature
install list of features
uninstalls the `<feature>` feature
Install DLUX, NETCONF, and RESTCONF

opendaylight_user@root> feature:install odl-dlux-core
opendaylight_user@root> feature:install odl-dluxapps-yangui
opendaylight_user@root> feature:install odl-restconf-all
opendaylight_user@root> feature:install odl-netconf-all
opendaylight_user@root> feature:install odl-netconf-topology
Opendaylight_user@root> feature:install odl-netconf-connector-ssh
opendaylight_user@root> feature:list -r

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Required</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>odl-netconf-topology</td>
<td>1.3.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-restconf-all</td>
<td>1.6.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-netconf-connector-ssh</td>
<td>1.3.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-dluxapps-yangui</td>
<td>0.6.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-netconf-all</td>
<td>1.3.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>odl-dlux-core</td>
<td>0.6.1</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>wrap</td>
<td>0.0.0</td>
<td>x</td>
<td>Started</td>
</tr>
<tr>
<td>standard</td>
<td>4.0.10</td>
<td>x</td>
<td>Started</td>
</tr>
</tbody>
</table>
http://localhost:8181/index.html#/yangui/index
Example Use Case
Mininet, OVSDB and OpenFlow
Honeycomb/VPP using NETCONF

• VPP is a high-performance, open source, software forwarder
  • http://www.fd.io

• Honeycomb provides NETCONF and RESTCONF interfaces to VPP
Cisco IOS XR using BGP-LS and PCE-P

- Cisco XRv topology in dCloud
  - dCloud is http://dcloud.cisco.com (requires CCO login)
  - “OpenDaylight Boron SR3 with Apps with 8 Nodes v1”
  - ODL runs in dCloud (or use anyconnect/openconnect VPN to use local ODL instance)
    - http://github.com/CiscoDevNet/open-daylight-setup
- Use Pathman-SR application to create Segment Routed LSPs
OpenDaylight with Mininet – Step by Step

• Install, setup, and start Mininet VM using VirtualBox
  • Great instructions at [http://www.brianlinkletter.com/set-up-mininet/](http://www.brianlinkletter.com/set-up-mininet/)
  • Login (user=mininet, password=mininet)

• Within OpenDaylight, enable required feature set
  • `opendaylight-user@root> feature:install odl-l2switch-switch odl-dlux-core odl-dluxapps-applications`

• Within Mininet VM, start 3 switches controlled by OpenDaylight
  • `mininet@mininet-vm:$ sudo mn --topo linear,3 --mac --controller=remote,ip=<OpenDaylight-IP>,port=6633 --switch ovs,protocols=OpenFlow13`
  • `mininet@mininet-vm:$ pingall`

• From browser, log into OpenDaylight DLUX
Mininet Network Start

```
[mininet@mininet-vm:~]$ sudo mn --topo linear,3 --mac --controller=remote,ip=192.168.40.18, port=6633 --switch ovs,protocols=OpenFlow13
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1 s2 s3
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (s2, s1) (s3, s2)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
[mininet]$ pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
mininet> 
```
Using DLUX

• From Browser, log into OpenDaylight DLUX
  • http://<OpenDaylight-IP>:8181/index.html
    (credentials: admin/admin)

• Check out the network and switches by clicking on Nodes, Node Connectors
REST APIs

- Click on **Yang UI** and **Expand All** to see the REST APIs available
Inventory of Network Nodes

- GET opendaylight-inventory -> operational -> nodes
Honeycomb/VPP Using NETCONF

Step by Step

1. Create VM for Honeycomb and VPP
2. Install VPP and Honeycomb on VM
3. Start VPP and Honeycomb
4. Connect to VPP using CLI
5. Add interface(s) to VPP
6. Connect to VPP using Honeycomb/NETCONF
7. Connect to VPP using OpenDaylight
Honeycomb/VPP Using NETCONF

1. Create VM for Honeycomb and VPP

   • Download minimal CentOS 7 from https://www.centos.org/download/

   • Create VM and enable ssh using http://www.jeramysingleton.com/install-centos-7-minimal-in-virtualbox/ to create VM and enable ssh
      • Add two host-only adapters with DHCP and promiscuous mode enabled
         • One for VPP, another to access Honeycomb directly from laptop
      • To add sudo for my user (devnet/devnet) using https://www.digitalocean.com/community/tutorials/how-to-create-a-sudo-user-on-centos-quickstart
Honeycomb/VPP Using NETCONF

2. Install VPP and Honeycomb on VM

- FD.io wiki provides instructions for installing VPP and installing HC
  - Add the FD.io repo:
    - Add the following lines to /etc/yum.repos.d/honeycomb-release.repo [honeycomb-release]
      name=honeycomb release branch latest merge
      baseurl=https://nexus.fd.io/content/repositories/fd.io.centos7/
      enabled=1
      gpgcheck=0
  - Install both packages
    - sudo yum install vpp
    - sudo yum install honeycomb
Honeycomb/VPP Using NETCONF

3. Start VPP and Honeycomb

• Important to start VPP first, then Honeycomb
  • sudo service vpp start
  • sudo service honeycomb start

• Check availability of Honeycomb’s SSH/NETCONF port:
  • netstat -an | grep 2831
  • You may have to clear iptables (Centos blocks most traffic by default)
    • iptables -P INPUT ACCEPT
    • iptables -P FORWARD ACCEPT
    • iptables -P OUTPUT ACCEPT
    • iptables -t nat -F
    • iptables -t mangle -F
    • iptables -F
    • iptables -X
Honeycomb/VPP Using NETCONF

4. Connect to VPP Using CLI

• Connect to VPP’s command line interface (CLI)
  • $ ssh devnet@192.168.60.101
  • $ sudo vppctl

```
$ ssh devnet@192.168.60.101
$ sudo vppctl
```

• $vpp# show interface

<table>
<thead>
<tr>
<th>Name</th>
<th>Idx</th>
<th>State</th>
<th>Counter</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>host-vpp1out</td>
<td>1</td>
<td>up</td>
<td>rx packets</td>
<td>6</td>
</tr>
</tbody>
</table>
Honeycomb/VPP Using NETCONF

5. Add interface(s) to VPP

- Add a virtual interface using [https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise:_Create_an_Interface](https://wiki.fd.io/view/VPP/Progressive_VPP_Tutorial#Exercise:_Create_an_Interface)

- Optionally add a physical NIC using [https://wiki.fd.io/view/VPP/How_To_Connect_A_PCI_INTERFACE_To_VPP](https://wiki.fd.io/view/VPP/How_To_Connect_A_PCI_INTERFACE_To_VPP)
  - Need to have associated a host-only network; if none, add one with DHCP and promiscuous mode before proceeding, should get something like
  - Details in notes section of slide
Honeycomb/VPP Using NETCONF

6. Connect to VPP Using Honeycomb and NETCONF

- Honeycomb listens on port 2831 for SSH/NETCONF
- Connect to VPP and issue for sample commands using: https://wiki.fd.io/view/Honeycomb/Releases/1609/Running_Honeycomb

- You also need to add ssh-dss when connecting via ssh
  - $ ssh -oHostKeyAlgorithms=+ssh-dss admin@192.168.60.101 -p 2831 -s netconf

- By default, honeycomb listens for RESTCONF on localhost:2831. To connect via RESTCONF from off-box
  - $ sudo vi /opt/honeycomb/config/restconf.json
    - Change restconf config from localhost or 127.0.0.1 to 0.0.0.0, e.g.
      "restconf-binding-address": "0.0.0.0",
      "restconf-port": 8183,
Honeycomb/VPP Using NETCONF

7. Connect to VPP Using OpenDaylight

- Enable NETCONF interface on OpenDaylight
  - feature:install odl-restconf-all odl-netconf-all odl-netconf-topology odl-netconf-connector-ssh

- Add VPP to OpenDaylight using Postman
  - PUT
    http://{odl_address}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1
  - Postman collection

- Interact with VPP using OpenDaylight DLUX
OpenDaylight as a Platform for Network Programmability
Conclusions
Key Takeaways

• SDN is more than just OpenFlow
• Network programmability is key benefit of SDN
• OpenDaylight provides a platform for network applications and programmable network infrastructure via YANG, NETCONF, RESTCONF
Additional resources
Open Source Dev Center

Your Source for Open Source at Cisco
https://developer.cisco.com/opensource

- Contributions to open source
- Use in products/solutions
- Community forums, blogs
- Developer Events
  - IETF Hackathons and MEF LSO Hackathons featuring open source implementations of open standards
OpenDaylight Microsite
https://developer.cisco.com/opendaylight
Building Applications on Top of OpenDaylight

- **AUTODEV**
  Visualize and manage IoT sensors embedded in motor vehicles

- **BGP and PCEP Pathman**
  Visualize topologies and program MPLS traffic engineering (TE) paths

- **BIERMAN**
  Visualize and manage BIER network topologies within ODL

- **DevNet Sample Apps**
  Learn how to use ODL and create your own apps that run on top of it

- **OpenFlow Manager**
  Visualize OpenFlow (OF) topologies, program OF paths, and gather OF stats

- **PCE-OpenFlow**
  Apply policy-based path computation traffic engineering to OpenFlow networks

- **YANG Explorer**
  Yang browser and RPC builder application to experiment with YANG models

- **In-band OAM (OAM)**
  Add operational info to packet as it traverses a path in network

- **VPP vBridge Manager**
  Define VPP-based virtual bridge domain(s) for L2 connectivity

- **YANGMAN**
  Dynamically generated UI forms and native JSON representation based on RESTCONF APIs

- **OneM2M Plugins**
  Extend the functionality of the OneM2M datastore. Protocol conversion, oneM2M data export are examples

- **OneM2M TSDR Plugin**
  Export oneM2M data to the OpenDaylight Time Series Data Repository

- **Pathman SR**
  Visualize topologies and program Segment Routing (SR) paths

- **Service Function Chaining**
  Create and deploy service chains using the NSH protocol as defined in draft-ietf-sfc-nsh

- **netACL**
  Program and manage Access Control Lists (ACLs) on routers in multi-vendor network
Tutorials and Sandboxes

OpenDaylight Nitrogen SR1 with Apps with 8 Nodes v1

Overview

OpenDaylight (ODL) is a collaborative, open-source project used to advance software-defined networking (SDN). OpenDaylight is a community-led, industry-supported framework consisting of code and blueprints. Using this framework, you can accelerate process adoption, foster innovation, reduce risk, and create a more transparent approach to SDN. OpenDaylight can be a core component within any SDN architecture. Building on open-source SDN and NFV controllers enables users to reduce operational complexity, extend the life of their existing infrastructure hardware, and enable new services and capabilities only available with SDN.

Scenarios

- Scenario 1: Explore ODL Features
- Scenario 2: Explore DLUX
- Scenario 3: Install BGP Pathman Application
- Scenario 4: Enable OpenFlow in Karaf
- Scenario 5: Install OpenFlow Manager Application
- Scenario 6: Explore Pathman Segment Routing
- Scenario 7: Explore netACL Application
- Scenario 8: Explore Yangman
OpenDaylight as a Platform for Network Programmability
Continue Your Education

• Become a DevNet Member:
  • https://developer.cisco.com/join

• Access OpenDaylight resources
  • https://developer.cisco.com/site/opendaylight/

• Visit our Open Source Dev Center:
  https://developer.cisco.com/site/opensource/
Thank you!