Move up the food chain with model driven network layer APIs

Charles Eckel, Cisco DevNet
eckelcu@cisco.com, @eckelcu
October 16, 2018
Agenda

• What is SDN
• Network programmability
• Network layer APIs
• 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

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
Network programmability
Why Network Programmability Matters

Network Expenses

<table>
<thead>
<tr>
<th></th>
<th>CAPEX</th>
<th>OPEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Forrester

Deployment Speed

- Computing
- Networking

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

NETCONF, RESTCONF and YANG

SNMP Experience

CLI Best Practices

Operations Requirements
YANG

Data Modeling Language for Networking

• Modeling language, defined in RFC 6020
• Models configuration and state data, RPCs, 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
YANG Model Example

- Screenshot from network-topology.yang
- Container 'network-topology' with list of 'topology' items
- List items (leafs) have a 'topology-id' which is also the key for the list

```yang
container network-topology {
  list topology {
    description "
      This is the model of an abstract topology.
      A topology contains nodes and links.
      Each topology MUST be identified by
      unique topology-id for reason that a network could contain many
      topologies.
    ";
    key "topology-id";
    leaf topology-id {
      type topology-id;
      description "
        It is presumed that a datastore will contain many topologies. To
        distinguish between topologies it is vital to have UNIQUE
        topology identifiers.
      ";
    }
    leaf server-provided {
      type boolean;
      config false;
      description "
        Indicates whether the topology is configurable by clients,
        or whether it is provided by the server. This leaf is
        populated by the server implementing the model.
        It is set to false for topologies that are created by a client;
        it is set to true otherwise. If it is set to true, any
        attempt to edit the topology MUST be rejected.
      ";
    }
  }
  container topology-types {
    description "
      ";
  }
```
Tools to work with YANG Models

- **pyang** - An extensible YANG validator and converter
  - Command line tool
  - 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 and RPC Builder
  - Web Based GUI
  - [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)
NETCONF
NETCONF
IETF network management protocol
• Defined in RFC 4741 (2006), updated by RFC 6241 (2011)
• Connection oriented, with transport via SSH/TSL
• Data defined by YANG models, encoded in XML
• Distinguishes between configuration and state data
• Multiple configuration datastores (candidate, running, startup)
• Change validation, transactions, filtering, and 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 and state data exposed as resources
- Access data using REST verbs (GET / PUT / POST …)
- Construct URIs, based on structure of YANG model, to access data
- HTTP instead of SSH for transport
- JSON in addition to XML for data encoding

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

http://localhost:8181/restconf/config/network-topology:
  - network-topology:
    - topology:
      - topology:
        - netconf/node/vpp1:
          <node xmlns="urn:TBD:params:xml:ns:yang:network-topology">
            <node-id>vpp1</node-id>
            <host xmlns="urn:opendaylight:netconf-node-topology">{{vpp1_address}}</host>
            <port xmlns="urn:opendaylight:netconf-node-topology">2831</port>
            <username xmlns="urn:opendaylight:netconf-node-topology">admin</username>
            <password xmlns="urn:opendaylight:netconf-node-topology">admin</password>
            <tcp-only xmlns="urn:opendaylight:netconf-node-topology">false</tcp-only>
            <keepalive-delay xmlns="urn:opendaylight:netconf-node-topology">0</keepalive-delay>
          </node>
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
  - NETCONF server
  - RESTCONF server
  - Manageability Infra
  - Config DB
  - BGP
  - QoS
  - VXLAN

- Manageability
  - Infra
  - Config
  - DB
  - YANG-based XML
  - ANY (C, Java, Python, Perl, PHP)
  - ANY (Java, Python, Perl, PHP)
  - RESTCONF client

- Manageability
  - Infra
  - Config
  - DB
  - YANG-based XML
  - ANY (C, Java, Python, Perl, PHP)
  - ANY (Java, Python, Perl, PHP)
  - RESTCONF client

- Manageability
  - Infra
  - Config
  - DB
  - YANG-based XML
  - ANY (C, Java, Python, Perl, PHP)
  - ANY (Java, Python, Perl, PHP)
  - RESTCONF client

© 2018 Cisco and/or its affiliates. All rights reserved. Cisco Public
Network Layer APIs
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 model cache
Access VPP via NETCONF and RESTCONF

• VPP is a high-performance software forwarder
• VPP programmable through CLI and low level API
• Honeycomb provides NETCONF and RESTCONF interfaces for VPP
• VPP and Honeycomb part of FD.io project
  • http://www.fd.io
Add VPP1

PUT

http://{odl_address}:8181/restconf/config/network-topology:network-topology/topology/topology-netconf/node/vpp1

Authorization

Content-Type
application/xml

Body

Cookies (1)

Name
jSESSIONID
Value
1ap882gt7pk1rgeo2pwm16
Domain
localhost
Path
/restconf
Expires
false
HTTP
false
Secure
false
# YangUI

## API History Collection Parameters

**ROOT**

- instance-identifier-patch-module rev.2015-11-21
- nc-notifications rev.2008-07-14
- netconf-node-topology rev.2015-01-14
- network-topology rev.2013-10-21
  - operational
    - network-topology
      - topology (topology-id)
        + topology-types
        + underlay-topology (topology-ref)
  + node (node-id)
    + link (link-id)
    + igp-topology-attributes
  + config
  + notifications rev.2008-07-14

### GET

```
/get/operational/network-topology:network-topology/topology/
    topology-netconf /node/ vpp1
```

Request sent successfully

<table>
<thead>
<tr>
<th>node list</th>
<th>node <a href="">node-id:vpp1</a></th>
</tr>
</thead>
<tbody>
<tr>
<td>node-id</td>
<td>vpp1</td>
</tr>
<tr>
<td>host</td>
<td>192.168.60.101</td>
</tr>
<tr>
<td>port</td>
<td>2531</td>
</tr>
<tr>
<td>connection-status</td>
<td>connected</td>
</tr>
</tbody>
</table>
Access CSR1000v using NETCONF and RESTCONF

• Cloud services router (CSR)
• Virtualized, run using Vagrant
• NETCONF and RESTCONF interfaces
• Access from OpenDaylight using NETCONF
  • $ ssh vagrant@127.0.0.1 -p 2223 -s netconf
  • Credentials are vagrant/vagrant
• Access from Postman using RESTCONF
Add XE node with XE cache directory

```json
1 -
2   "network-topology:node": {
3     "node-id": 
4       "{{router}}",
5     "host": 
6       "{{router-ip}}",
7     "port": 
8       "{{router-port}}",
9     "username": 
10    "{{router-user}}",
11   "password": 
12     "{{router-pass}}",
13     "tcp-only": false,
14     "keepalive-delay": 0,
15     "schema-cache-directory": "XE"
16   }
```
Configure a Link between VPP and CSR1000v
Configure GigE on CSR
Configure GigE on VPP

```json
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
{"interface": [
    {"name": "GigabitEthernet0/8/0",
     "enabled": true,
     "description": "link to CSR",
     "type": "lan-1-if-type:ethernetCsmacd",
     "ietf-ipv4": {
     "address": ["192.168.1.2",
                "netmask": "255.255.255.0"
    ]
    }
],
"v3po:ethernet": {
"mtu": 9216
}]
```
Configure a Link between VPP and CSR1000v

- POSTMAN
- Network Layer APIs
- RESTCONF
- NETCONF Topology
- Model Cache
- MD-SAL
- OPEN DAYLIGHT

VPP

CSR1000v
The industry is referencing the LSO framework for next-gen OSS & the foundation for autonomous networking.
Business Applications View

LSO Cantata and Sonata (East-West)
Service Level
- Address Qualification
- Serviceability
- Quote
- Ordering
- Ticketing

LSO Legato and Presto (North-South)
Network Level
- Configuration and Activation
Service View

- Service Orchestration in each operator domain to orchestrate an IP ‘service’
  - Phase 1
    - Hong Kong to New Jersey
  - Phase 2
    - New Jersey to South Africa
    - New Jersey to Uganda
Network View

- Network Resource Provisioning in each operator domain
- LSO Legato and Presto
Conclusions
Key Takeaways

• SDN is more than just OpenFlow
• Network programmability is key benefit of SDN
• Network layer APIs unlock true potential of programmable network
Helpful Links

• Join DevNet: [https://developer.cisco.com/join/iitrtc18](https://developer.cisco.com/join/iitrtc18)
• Cisco DevNet: [https://developer.cisco.com](https://developer.cisco.com)
• Learning Labs: [https://learninglabs.cisco.com](https://learninglabs.cisco.com)
• Sandboxes: [https://developer.cisco.com/site/sandbox/#](https://developer.cisco.com/site/sandbox/#)
• @eckelcu: [https://twitter.com/eckelcu](https://twitter.com/eckelcu)
Thank you!