Google

Model-driven network management

Anees Shaikh, Josh George
Google Network Operations / OpenConfig operator working group

OPENCONFIG
www.openconfig.net

Zero-touch networking : extreme automation

- network operations fully automated based on operator-expressed intent
- introducing new technologies is transparent to the operator interface
- automated detection of unintended behavior and rollback
- automation complies with network policies for safety and security

Where operators would like to get to

scalable, real-time monitoring -- push not poll, pub-sub, incremental updates

declarative, intent-driven configuration -- what not how, abstractions

rapid deployment of new technologies -- dramatically shorten qual cycle

leverage platforms instead of scripts -- software skills, fewer CLI jockeys

policy compliant operations -- scoped operations, safety checks

What's holding them back

- proprietary integrations
  - CLIs, scripts, templates, modules, cookbooks, minions, …
- lack of abstractions and common APIs
  - CLI scraping from devices
- SNMP monitoring -- poor scaling, lack of extensibility, proprietary MIBs
- complexity and cost have been pushed to operators
  - long qualification processes, specialized skills, ...

Value of data models

- durable APIs for managing and monitoring the network
- management abstraction layer (insulate from low level details)
- forward compatibility with new platforms and technologies
- establishes a contract between NMS and infrastructure

OpenConfig's early focus is on defining industry-wide vendor-neutral APIs for network configuration and monitoring

OpenConfig : user-defined models

- informal industry collaboration among network operators
- data models for configuration and operational state, code written in YANG
- organizational model: informal, structured like an open source project
- development priorities driven by operator requirements
- engagements with major equipment vendors to drive native implementations
- engagement with standards (IETF) and OSS (ODL, ONOS, goBGP, Quagga)


Rest of the talk ...

re-architecting network monitoring and visibility

model-based configuration management

update on OpenConfig -- models, implementations, operationalizing

OpenConfig integration with OSS projects incl. ODL

Modern Network Telemetry

Warning: If you are sensitive to graphic depictions of SNMP, please leave the room.

SNMP needs an upgrade

legacy implementations -- designed for limited processing and bandwidth

expensive discoverability -- re-walk MIBs to discover new elements

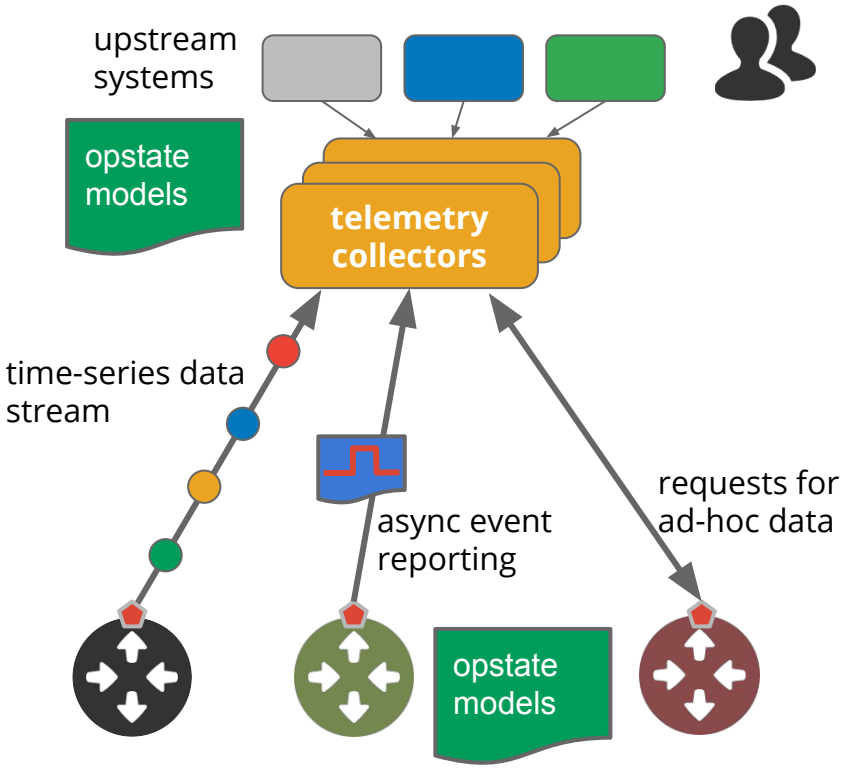
no capability advertisement -- test OIDs to determine support

rigid structure -- limited extensibility to add new data

proprietary data -- require vendor-specific mappings and multiple requests to reassemble data

protocol stagnation -- no absorption of current data modeling and transmission techniques

Streaming telemetry -- moving forward from SNMP



- stream data continuously -- with incremental updates
- telemetry sent based on subscriptions
- observe network state through a time-series data stream
- device data follows a common model
- efficient, secure transport protocols (gRPC)

Aim to deprecate SNMP in our network by 2017

gRPC : an open, multi-platform RPC framework

gRPC leverages HTTP/2 as its transport layer

- binary framing, header compression
- bidirectional streams, server push support
- connection multiplexing across requests and streams

gRPC features

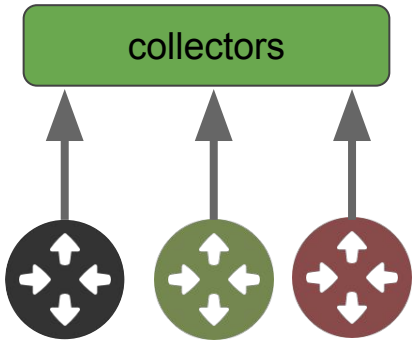
- load-balancing, app-level flow control, call-cancellation
- serialization with protobuf (efficient wire encoding)
- multi-platform, many supported languages -- incl. Maven plugin
- open source, under active development

@grpcio

Toward model-based streaming telemetry

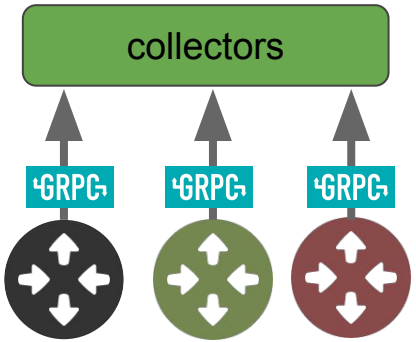
Step 1 -- from poll to push

proprietary data over proprietary transport, partial coverage



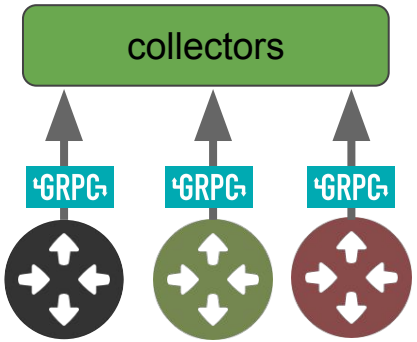
Step 2 -- more complete data over RPC channel

proprietary data over gRPC transport, increased coverage



Step 3 -- common data model over RPC

gRPC transport with common schema



Models structured for observing network behavior

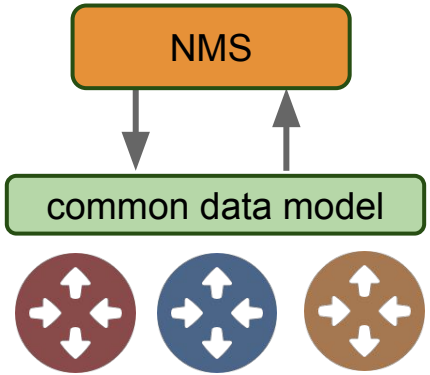
intended v. applied config

```
+--rw interface* [name]
   +--rw config
   |  +--rw type
   |  +--rw mtu?
   |  +--rw name?
   |  +--rw description?
   |  +--rw enabled?
   +--ro state
   |  +--ro type
   |  +--ro mtu?
   |  +--ro name?
   |  +--ro description?
   |  +--ro enabled?
   |  +--ro oper-status
   |  +--ro last-change?
   |  +--ro counters
   |     +--ro in-octets?
   |     +--ro in-unicast-pkts?
   |     +--ro in-broadcast-pkts?
   |     +--ro in-multicast-pkts?
   |     +--ro in-discards?
   |     +--ro in-errors?
   |     +--ro in-unknown-protos?
   |     +--ro out-octets?
```

opstate = applied configuration + derived stated (counters, etc.)

data models structured to ease programmatic access to related operational state

opstate has deterministic, explicit location in the model paths

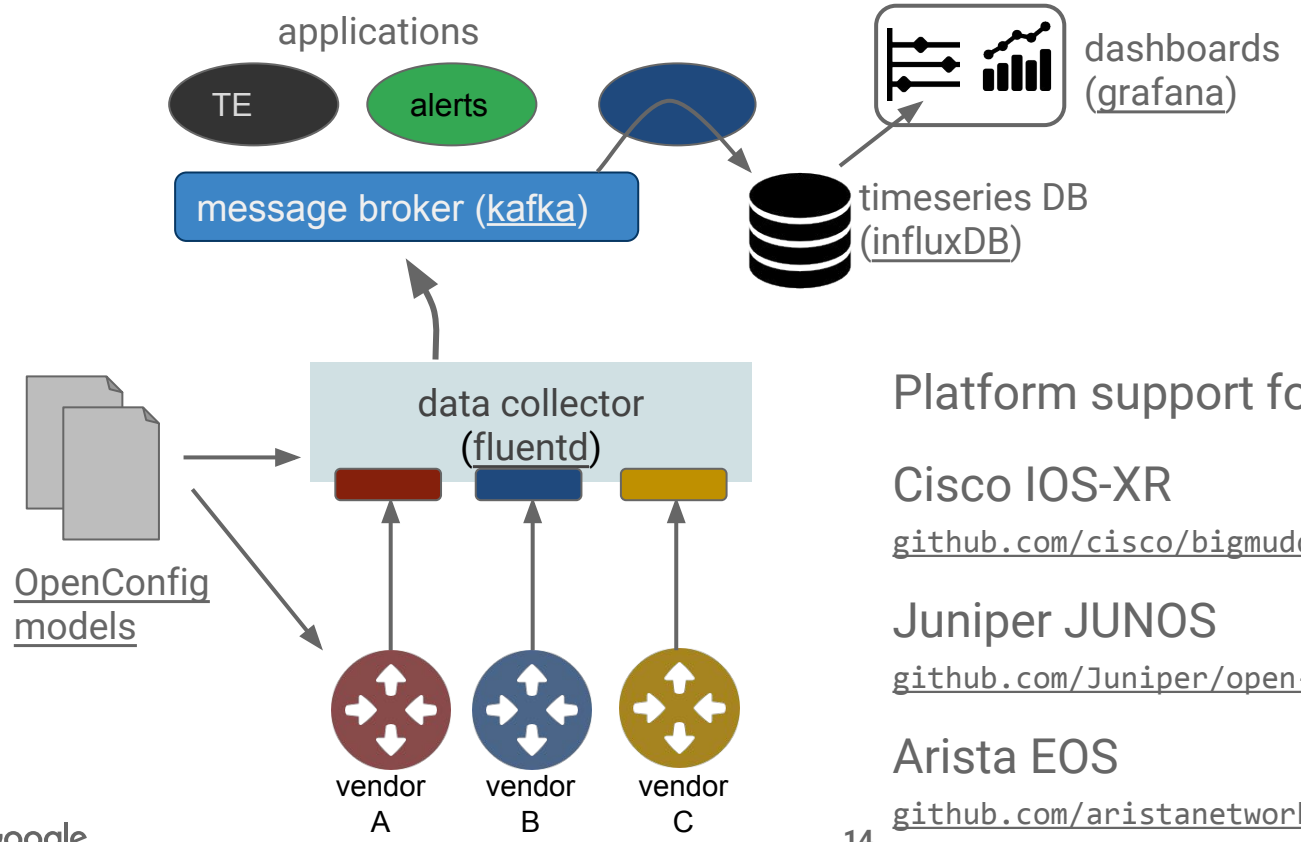


NMS acts on streaming view of state

quickly determine discrepancy between intended and applied configuration

code to exploit this pattern in multiple operators

OSS stack for model-based streaming telemetry



Platform support for streaming telemetry:

Cisco IOS-XR

github.com/cisco/bigmuddy-network-telemetry-stacks

Juniper JUNOS

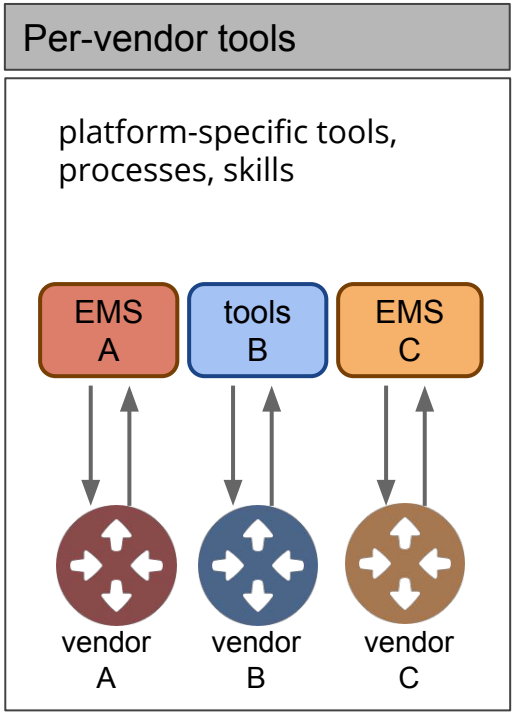
github.com/Juniper/open-nti

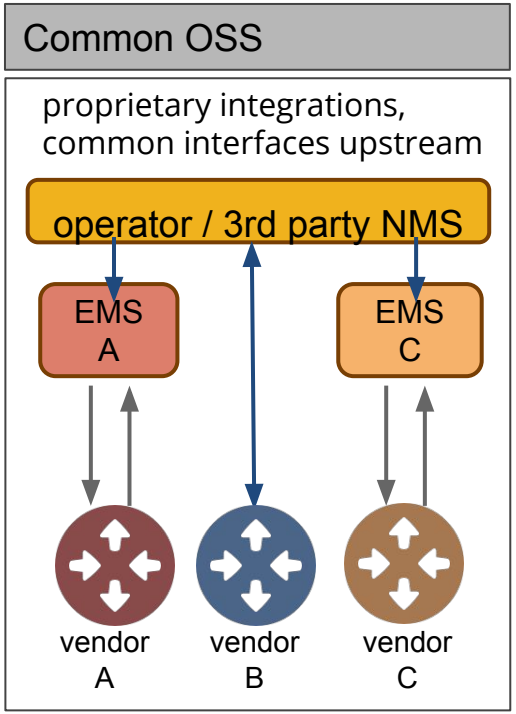
Arista EOS

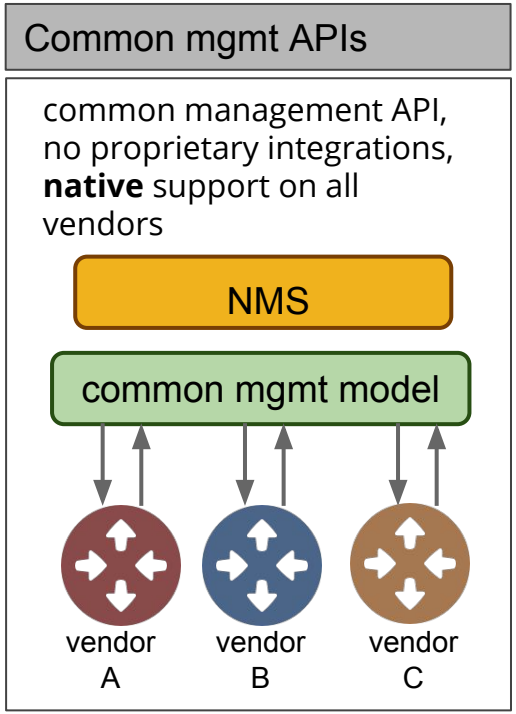
github.com/aristanetworks/goarista

Declarative Network Configuration

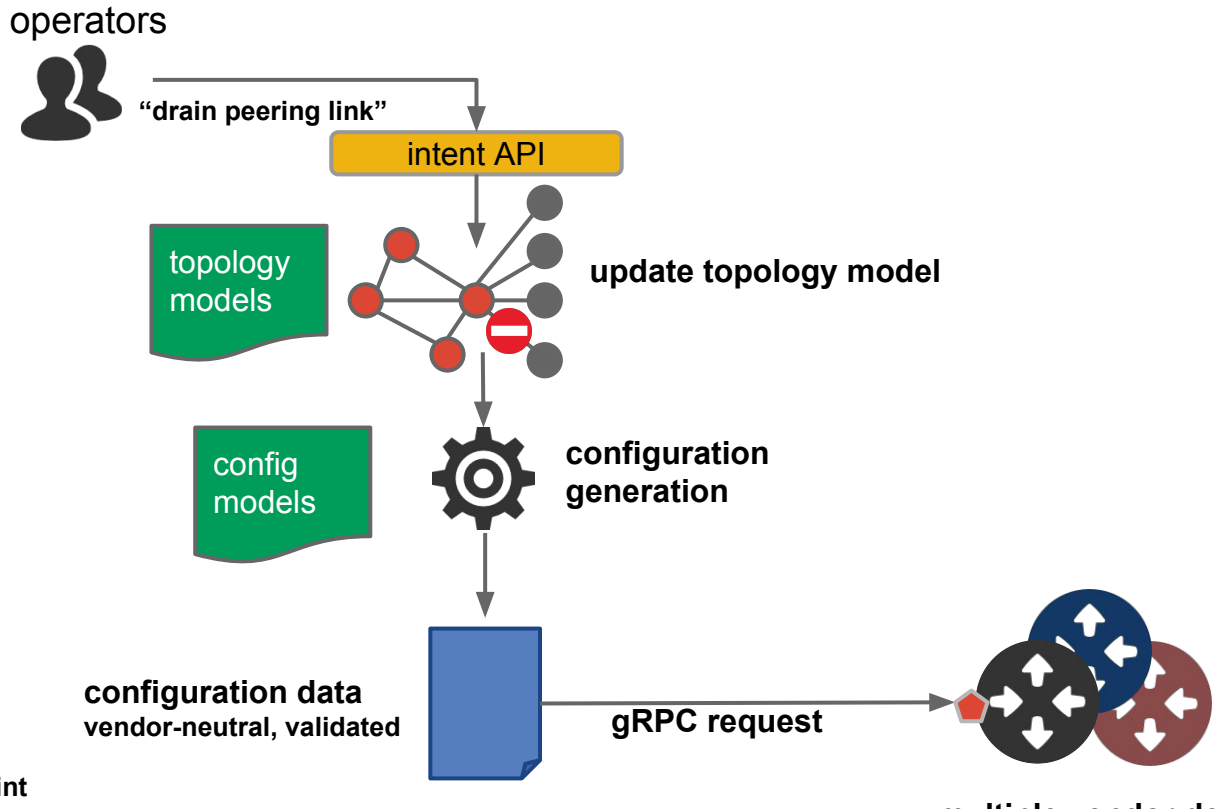
Toward a vendor-neutral, model-driven world



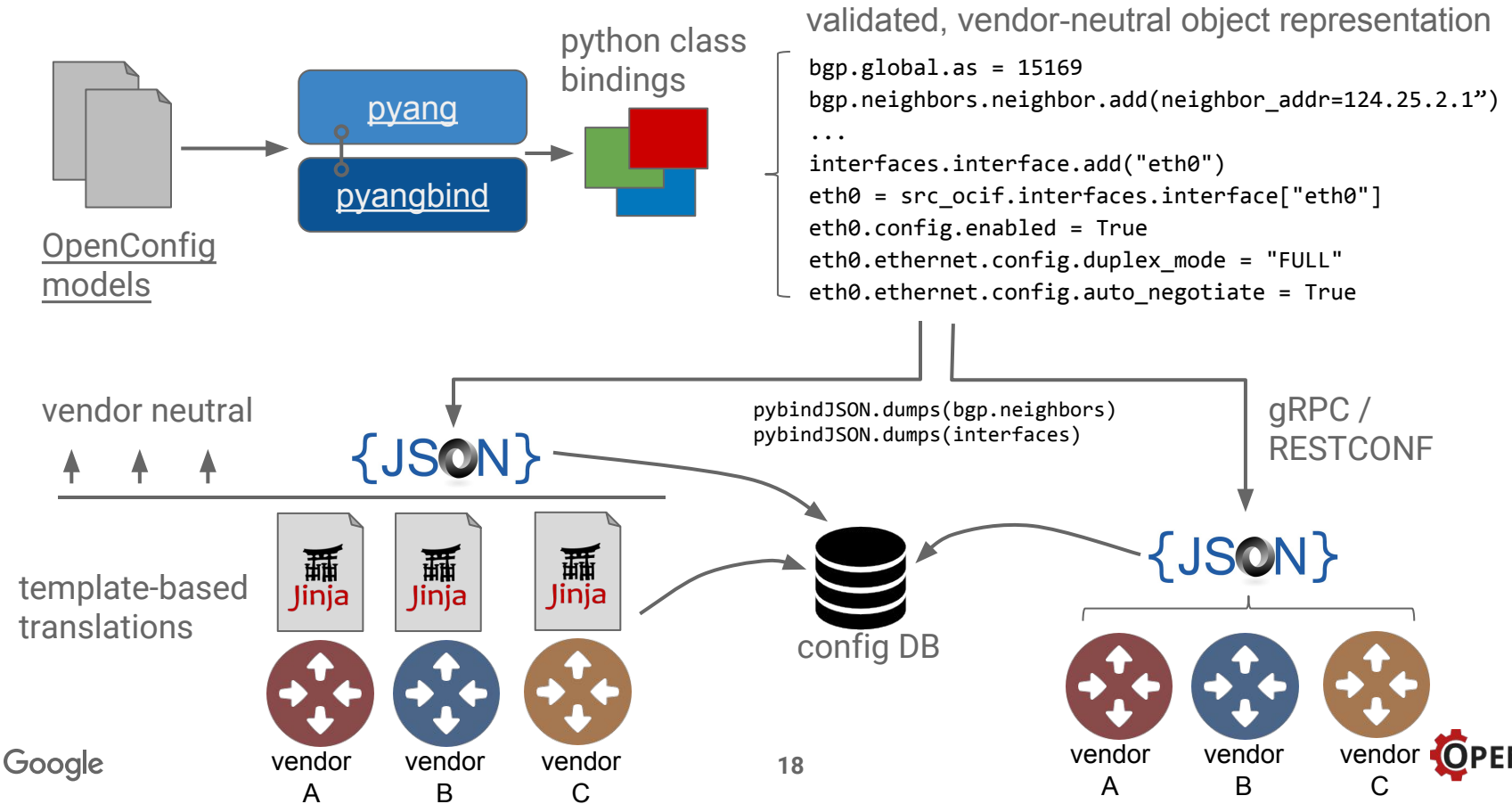




A model-based configuration pipeline



OSS stack for model-based programmatic configuration



OpenConfig

OpenConfig progress

Published OpenConfig models

- BGP*
- routing policy*
- locally generated routes*
- interfaces, aggregates, IP, VLANs*
- MPLS, RSVP / TE*
- networking / forwarding instances, VRFs
- RIB contents
- terminal optics*
- streaming telemetry configuration*
- top-level device structure

Models adopted by IETF for standardization
* native platform implementations available or in progress

Models in development / review

- system inventory / hardware
- ACLs
- line optics (EDFAs and ROADMs)
- system management
- IS-IS
- QoS
- tunnels / encapsulation
- LLDP
- FIB / LFIB

Additional publications

- Reference RPC specification
- YANG authoring style guide
- Guidelines for versioning models

OpenConfig and telemetry support on vendor devices

- initial versions of streaming telemetry available in 2016
  - Cisco Streaming Telemetry (IOS-XR)
  - Juniper JUNOS Telemetry Interface
  - additional vendors in progress
- OpenConfig BGP+policy model configuration native support
  - Cisco IOS-XR
  - Juniper JUNOS
  - Arista EOS
  - additional vendors with implementations underway
- implementations in progress for interfaces, MPLS / TE models, terminal optics
- operational state models being delivered as part of streaming telemetry

OpenConfig support in OSS projects

OpenDaylight BGP

- BGP, policy, local-routing, network-instance

Quagga BGP -- OpenConfig support in progress

Zebra 2.0

- OpenConfig BGP (see NetDev 1.1 presentation)

OpenConfig tooling available today

language bindings / data serialization

pyangbind -- Python classes from YANG models, JSON and XML serialization

goyang -- Go language compiler for YANG models

telemetry collectors

Go language gRPC collector -- available soon

BigMuddy -- Cisco UDP telemetry collector

OpenNTI -- Juniper UDP telemetry collector

Arista gRPC telemetry collector

YANG model authoring

OpenConfig style guide and checker

AT&T YANG design studio

NMS client / server

pynms -- (beta) gRPC client / server for sending configuration and telemetry collection

Tour of OpenConfig models

BGP and routing policy -- most mature, several implementations

telemetry configuration -- setting up streaming telemetry

platform -- device inventory and platform specifics

rpc -- protocol-independent spec of an OpenConfig client/server API

Challenges in thinking “model-driven”

CLI habits are hard to break

implementations are often very different (occasionally for a good reason)

models for machines or humans

what to leave out of the model

how to test for compliance by an implementation

Model-based automation in practice

- Operationalizing models -- many problems to be solved
  - protocol-agnostic RPC specification for encoding, delivering, and receiving data
  - handling mixed-schema operation (OpenConfig + native)
  - model updates and versioning in the NMS and on devices
- Engaging with vendors
  - this is all new for vendors -- close engineering engagements required
  - must have their own internal model; decouple model iterations from major releases
  - investment in infrastructure to support externally defined models is critical
- Tools lacking but being developed by users and vendors

OpenDaylight NMS based on OpenConfig

- ODL already supports some management / operations features
  - monitoring and path management: SNMP, BGP-LS / PCEP
  - network configuration: NETCONF / RESTCONF, OVSDB
  - data management and modeling: YANG tools, time-series data repo
- Potential additional capabilities
  - streaming telemetry collector, with pub/sub
  - support for additional data transports and encodings, e.g., gRPC
  - configuration validation

Summary

OpenConfig operator group is developing and publishing data models for monitoring and configuration -- built by users for users

Growing support in vendor implementations and OSS projects

- streaming telemetry and configuration

Range of open source tooling exists to build NMSes supporting OpenConfig

- great use case for OpenDaylight and other projects