Google Cloud

OpenConfig and telemetry overview for A-CORD

Anees Shaikh (Google)
on behalf of Google network operations and OpenConfig group

OPENCONFIG
www.openconfig.net

Agenda

OpenConfig project overview

Streaming telemetry

Open discussions on applications for A/CORD

OPENCONFIG

Projects

Data models

models for common configuration and operational state data across platforms

Streaming telemetry

Scalable, secure, real-time monitoring with modern streaming protocols

RPCs and tools

Management RPC specs and implementations

Tooling to build config and monitoring stacks

Participants

Google at&t Microsoft BT facebook Level(3) COMMUNICATIONS verizon YAHOO! COMCAST

COX JIVE T · · TeraStream Bell SK telecom Bloomberg NETFLIX ORACLE CLOUDFLARE

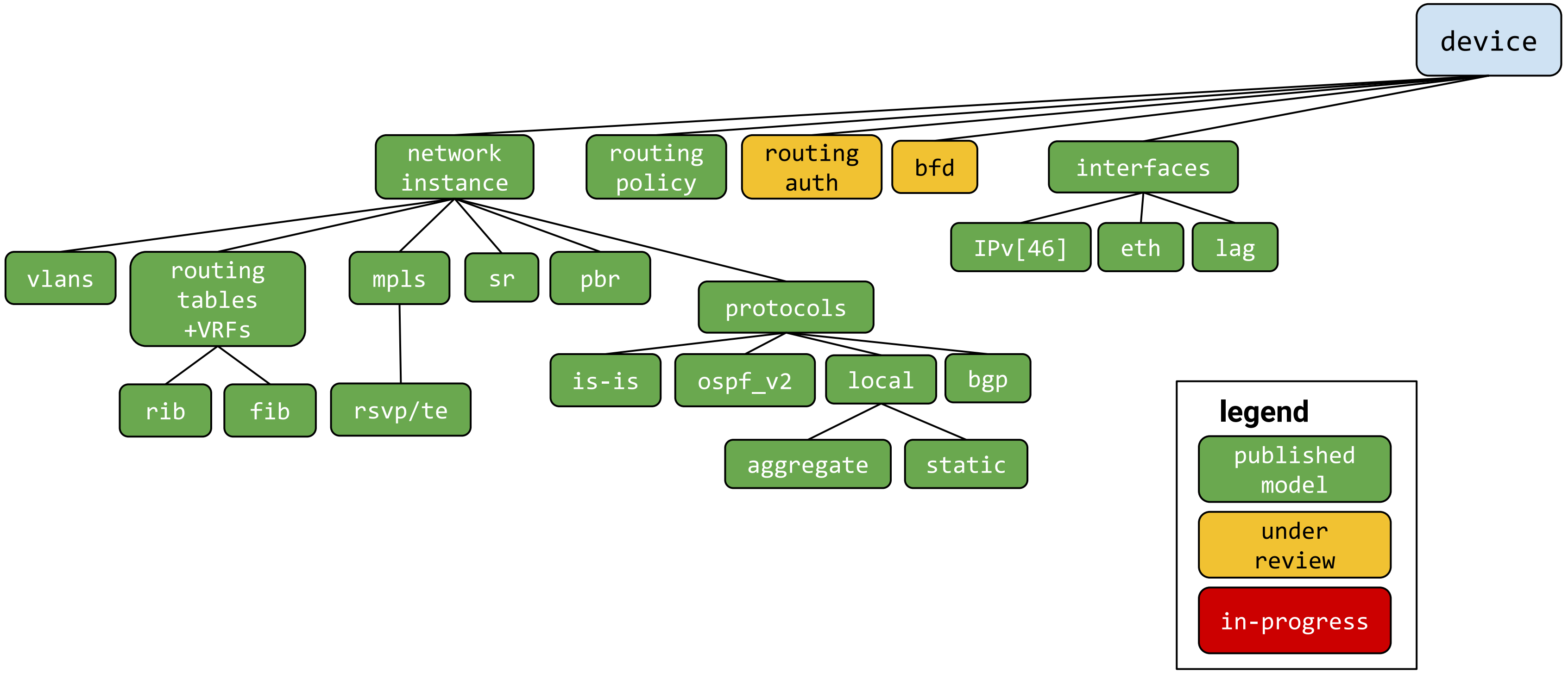
OpenConfig data models

- data models for configuration **and** operational state, written in YANG
- initial focus: device data for switching, routing, and transport (WiFi coming soon)
- development priorities driven by operator requirements
- technical engagement with major vendors to deliver native implementations

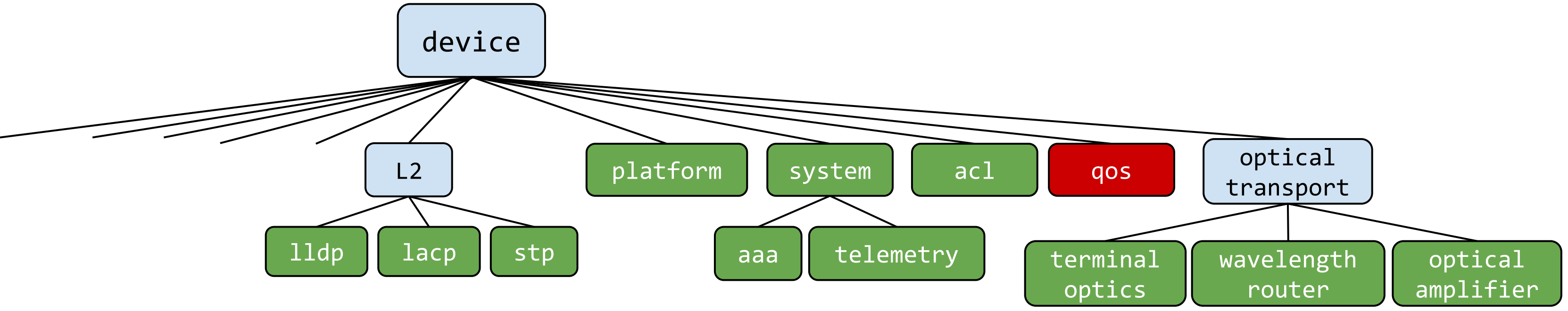
Implementations shipping or in-progress from multiple vendors

ARISTA ciena CISCO infinera JUNIPER NETWORKS NOKIA

OpenConfig data model progress 1/



OpenConfig data model progress 2/



legend

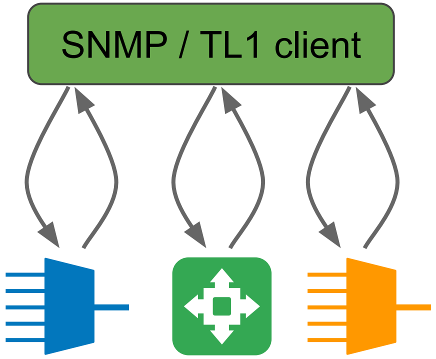
- 🟩 published model
- 🟨 under review
- 🟥 in-progress

Better visibility with streaming telemetry

operational state monitoring is crucial for network health and traffic management

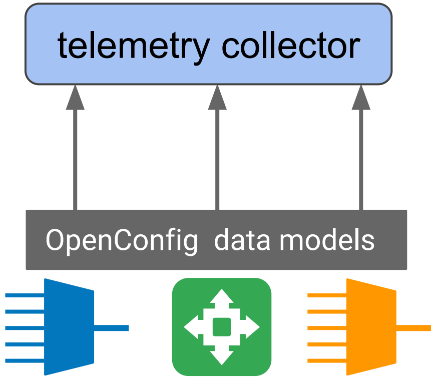
- counters, power levels, protocol stats, up/down events, inventory, alarms, ...

SNMP/ TL1 POLLING



- O(min) polling
- resource drain on devices
- legacy implementation
- inflexible structure
- proprietary structure

STREAMING TELEMETRY



- subscribe to desired data based on models
- streamed directly from devices
- time-series or event-driven data
- modern, secure transport

Production deployments of streaming telemetry from multiple vendors

Elements of a streaming telemetry solution

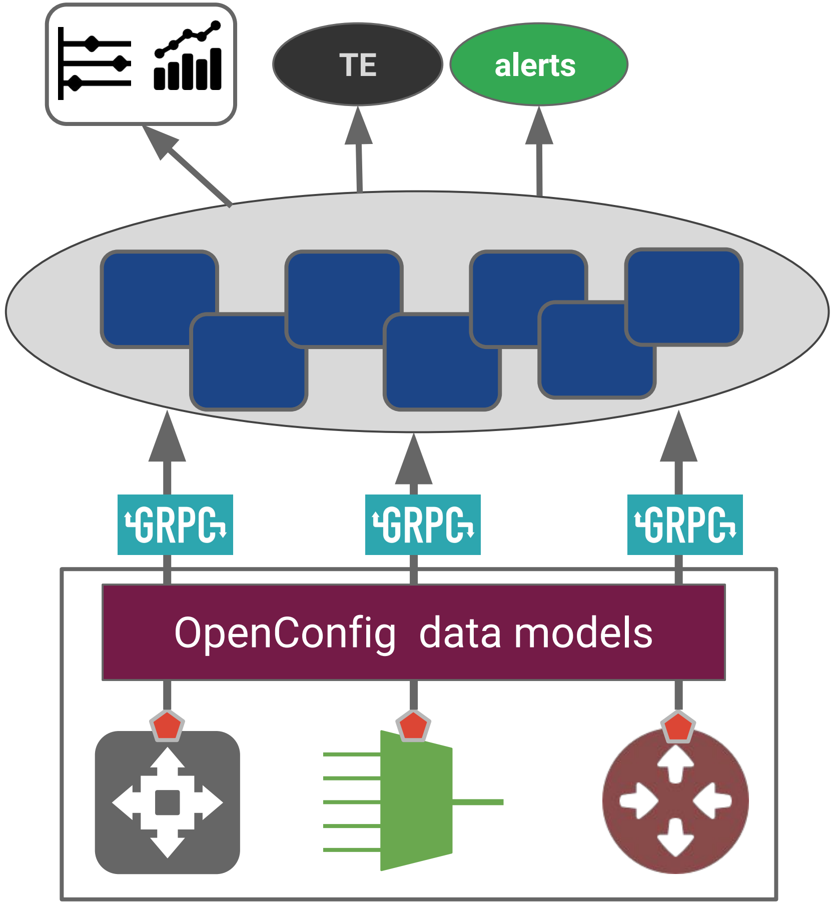
analytics applications

scalable collector infrastructure

common streaming protocol

data normalization

device instrumentation



Realized benefits of streaming telemetry

Production deployments on multiple routing and transport platforms

better data coverage -- 2-3x number of variables

higher frequency -- fresher data for automation, health-checking, and control

event-driven notifications -- faster reaction and recovery

reliable delivery w/TCP (vs. SNMP w/UDP)

normalized data based on common data models

gNMI -- management software built on gRPC

gRPC -- performant, secure RPC framework evolved from Google Stubby

- bidirectional streaming built on standard HTTP/2
- pluggable load balancing, tracing, health checking and auth
- client libraries in 10 languages

@grpcio
v. 1.x

gNMI -- gRPC Network Management Interface

- single service for state management (streaming telemetry and configuration)
- offers an implemented alternative to NETCONF, RESTCONF, …
- designed to carry any tree-structured data (not only YANG-modeled)

Objective-C

OpenConfig tools ecosystem

language bindings / data serialization

pyangbind -- Python classes from YANG models, JSON serialization

goyang -- Go language compiler for YANG models

OpenConfig Go library -- library to create and validate config instances (internal)

YANG model authoring

OpenConfig style guide

OpenConfig YANG model checker

OpenConfig documentation generator

telemetry collectors

Go language gNMI collector reference impl

BigMuddy -- Cisco UDP telemetry collector

OpenNTI -- Juniper UDP telemetry collector

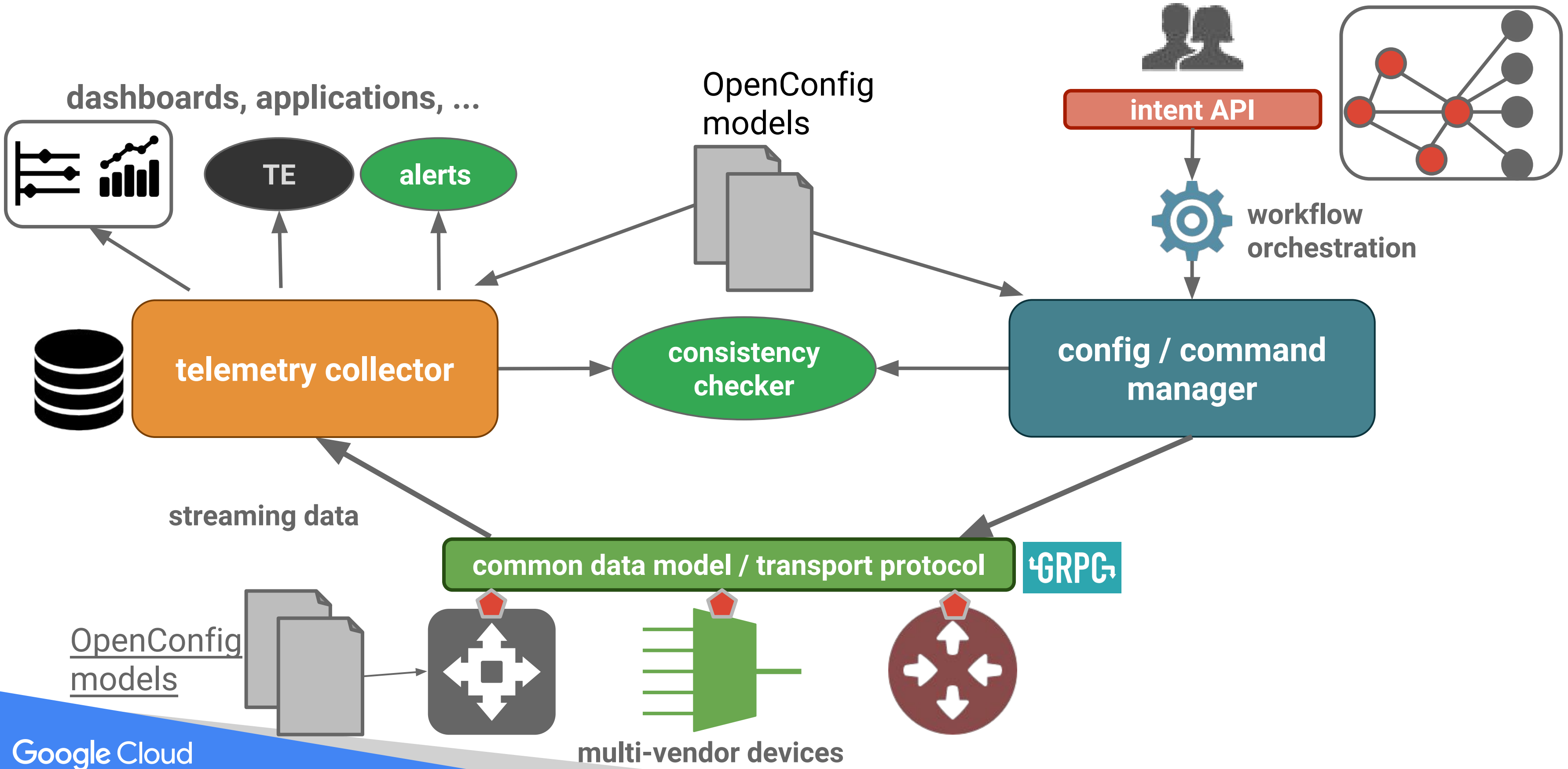
Arista -- gRPC telemetry collector

NMS client / server

gNMI -- gRPC based management protocol spec

pynms -- example Python NMS code (beta)

Putting it all together



Engaging with OpenConfig

network operators

- just join -- bring use cases, model extensions, tools, reviews, …
- use the models and tools -- help improve them
- push your vendors for native support

vendors

- feedback on models (particularly on implementability)
- implement streaming telemetry and native model support
- engage via your customers

OSS projects and ISVs

- adopt OpenConfig as a management API for common elements
- continue to build the model-based management ecosystem

Thank you

Anees Shaikh
(with contributions from many in Google networking and the OpenConfig working group)

aashaikh@google

Extensions to gNMI

current gNMI definition supports only NMS-initiated connections to target devices

- extend to “dial-out” to support target-initiated connections

new services for operational commands

- e.g. ping, traceroute, reboot, clear BGP session, update firmware, …
- considering as a set of microservices , separate from main gNMI service

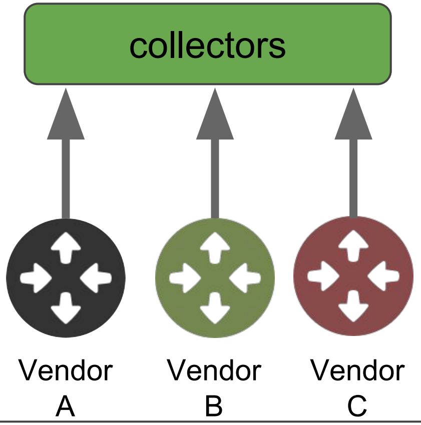
native Protobuf value encoding

- avoid type-casting to strings during encoding

Toward model-based streaming telemetry

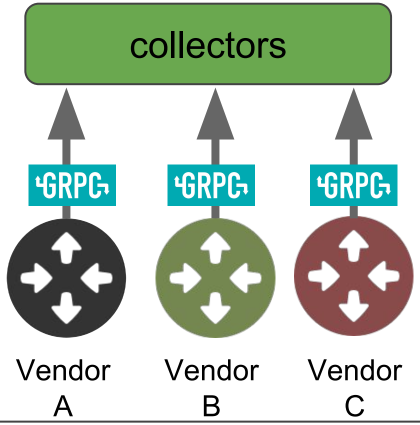
Step 1 -- from poll to push

proprietary data over proprietary transport, partial coverage



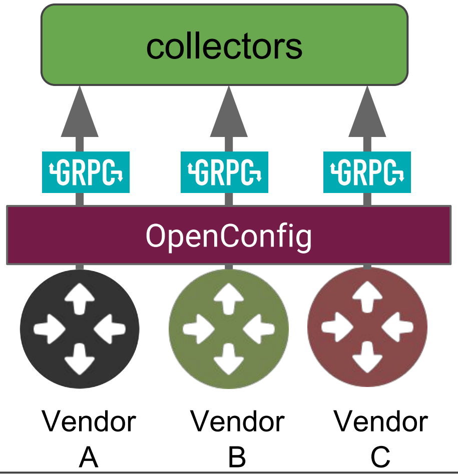
Step 2 -- more complete data over RPC channel

proprietary data over gRPC transport, increased coverage



Step 3 -- common data model over RPC

gRPC transport with common schema



Google Cloud

The gNMI service

```
option (gnmi_service) = "0.2.2";
service gNMI {
 // Retrieve the set of capabilities supported by the target.
 rpc Capabilities(CapabilityRequest) returns (CapabilityResponse);

 // Retrieve a snapshot of data from the target.
 rpc Get(GetRequest) returns (GetResponse);

 // Modify the state of data on the target.
 rpc Set(SetRequest) returns (SetResponse);

 // Subscribe to stream of values of particular paths within the data tree.
 rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
```