Better management of large-scale, heterogeneous networks
toward a programmable management plane

Joshua George, Anees Shaikh
Google Network Operations

www.openconfig.net
Agenda

1. Management plane challenges
2. Rethinking telemetry -- efficient, large-scale monitoring
3. OpenConfig -- community-driven API development
Management Plane Challenges
Challenges of managing a large-scale network

- 20+ network device roles
- more than half dozen vendors, multiple platforms
- 4M lines of configuration files
- up to ~30K configuration changes per month
- more than 8M OIDs collected every 5 minutes
- more than 20K CLI commands issued and scraped every 5 minutes
- many tools, and multiple generations of software

*Opportunity for significant OPEX savings*: reduced outage impact, simplification of management stack, better scaling
Management plane is way behind

- proprietary CLIs, lots of scripts
- imperative, incremental configuration
- lack of abstractions
- configuration scraping from devices
- SNMP monitoring -- not always “simple” and not often scalable
Model-driven network management

**Topology**
- describes structure of the network
- common modeling language: multiple
- data encoding: protobuf, ...

**Configuration**
- describes configuration data structure and content
- common modeling language: YANG
- multiple data encodings: protobuf, XML, JSON, ...

**Telemetry**
- describes monitoring data structure and attributes
- common modeling language: exploring YANG
- data delivery: RPC, protobuf inside UDP
Rethinking Network Telemetry
Telemetry solutions today

What do we use? Often SNMP is the default choice.

- 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
Telemetry challenges

- SNMP object collection growing with each platform generation
  - e.g., 100K objects on current platforms, expected to grow 3x over next 2 generations
  - similar for object collection frequency
- Future devices continue to grow in density and drive this trend
  - scale limitations in data acquisition at high frequencies
- Near-real-time acquisition and access to monitoring data is a requirement for <insert buzzword here>
  - traffic management, tight control loops, fast recovery
I get it, you really don’t like SNMP...

but do you have a better idea?
Rethinking telemetry... reverse the flow

- stream data continuously -- with incremental updates based on subscriptions
- observe network state through a time-series data stream
- devices programmed with a data model describing desired structure and content
- efficient, secure transport protocols
Telemetry framework requirements

- network elements stream data to collectors (push model)
- data populated based on vendor-neutral models whenever possible
- utilize a publish/subscribe API to select desired data
- scale for next 10 years of density growth with high data freshness
  - other protocols distribute load to hardware, so should telemetry
- utilize modern transport mechanisms with active development communities
  - gRPC (HTTP/2), Thrift, etc.
  - protocol buffer over UDP
Example telemetry configuration flow

Structured inventory and set of telemetry capabilities pushed to the NMS

Network Management System

Generate monitoring configuration

Publish to network element

gRPC endpoint

Rules for typical monitoring

Tactical overrides from operators
3 types of telemetry events:
- Bulk time series data
  - All interface stats every 10 seconds.
- Event/edge driven updates
  - LSP A is now down.
- Operator request/response
  - Show me operator state for all interfaces.
Practical realization

- Streaming telemetry is beyond an idea stage, but is far from a final product
- Multiple vendor implementations now available for experimentation
- Development is ongoing -- now is the time to share your requirements and make your voice heard!
OpenConfig
OpenConfig

- Informal industry collaboration of network operators
- Focus: define vendor-neutral configuration and operational state models based on real operations
  - Adopted YANG data modeling language (RFC 6020)
- Participants: Apple, AT&T, BT, Comcast, Cox, Facebook, Google Level3, Microsoft, Verizon, Yahoo!
- Primary output is model code, published as open source via public github repo
- Ongoing interactions with standards and open source communities (e.g., IETF, ONF, ODL, ONOS)
Example configuration pipeline

operators

“drain peering link”

intent API

update topology model

configuration generation

configuration data
vendor-neutral, validated

gRPC req
gRPC endpoint

OC YANG models

devices
Extending OpenConfig models

- base OpenConfig model as a starting point
- vendors can offer augmentations / deviations
- operators can add locally consumed extensions
OpenConfig releases and roadmap

Data models (configuration and operational state)

- BGP and routing policy
  - multiple vendor implementations in progress
- MPLS / TE consolidated model
  - RSVP / TE and segment routing model as initial focus
- design patterns for operational state and model composition
- tools for translating YANG models to usable code artifacts
  - e.g., pyangbind

Models in progress
- interfaces, system, optical transport, ...
Models must be composed to be useful

- model composition framework is critical missing piece from existing model-building efforts
Modeling operational state

Types of operational state data

- derived, negotiated, set by a protocol, etc. (negotiated BGP hold-time)
- operational state data for counters or statistics (interface counters)
- operational state data representing intended configuration (actual vs. configured)

Clear benefits from using YANG to model both configuration and operational state in the same data model

- but ... YANG focus has primarily been config, NETCONF-centric, lack of common conventions
Summary

- New networking paradigms like SDN focus mostly on control
  - it’s time for the management plane to join the age of SDN

- **Core principles:**
  - model-driven management
  - streaming telemetry to scale monitoring and improve freshness
  - vendor-neutral, extensible APIs for managing devices

- Architecture and emerging vendor implementations of multi-mode telemetry solutions

- OpenConfig is a focused effort by operators to develop vendor-neutral models to define management APIs

*Operators: get involved and push your vendors for support on your gear!*
thank you !
gRPC: multi-platform RPC framework

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

gRPC leverages HTTP/2 as its transport layer
- binary framing, header compression
- bidirectional streams, server push support
- connection multiplexing across requests and streams
Additional “observations”

- YANG and NETCONF should be decoupled -- each are independently useful
- YANG needs to evolve more rapidly at this early phase, stabilize as real usage increases
- current YANG model versioning is not helpful -- treat models like software artifacts, not dated documents
- current standard models should be open for revisiting and revising
- should not rush to standardize more models until they are deployed and used in production

these are not necessarily OpenConfig consensus views
Current OpenConfig “process”

- initial models developed by OpenConfig
- extensive collaboration with vendors
- leverage existing work where possible
- publish models and docs
Intent-based configuration flow

- configuration intent
- abstract configuration models
  - Config Model
  - Topology Model
- declarative API
- device-level configuration
  - standard models
  - config generation
  - authoritative config store
- configuration pusher
  - NETCONF, RESTCONF, JSON-RPC, ...

analagous SDN stack

application

NB APIs

Network OS

SB protocols

operators

vendor-neutral configuration models

generated configuration instances