Network management at scale with gRPC

Anees Shaikh, Rob Shakir
on behalf of Google Networking
Elements of network management systems

- workflows, users
- configuration generation
- command execution
- operational state collector
- control API
- network devices
- monitoring, control systems
- traffic engr
State-of-the-art: monitoring

Operational state monitoring is crucial for network health and traffic management:
- counters, power levels, protocol stats, up/down events, inventory, alarms, ...

SNMP/T1 POLLING

- Lack of device timestamps
- Connectionless-transport
- Limited to O(min) polling
- Resource drain on devices
- Legacy implementations
- Proprietary structure

Persistent SNMP timeouts
Monitoring based on streaming telemetry

**STREAMING TELEMETRY**

- telemetry collector

  - collector subscribes to desired data
  - streamed directly from devices
  - time-series or event-driven data

**REQUIREMENTS**

- streaming data channels
- source timestamps on every item
- efficient delivery of high volume data
- reliable delivery of events
- secured against spoofing, MiTM, etc.
- portable to multiple device platforms
Streaming telemetry implementation options

**WebSockets**
- bidi communication, integration with HTTP
- favored by some vendors
- limited adoption

**gRPC, Thrift**
- streaming support, reliable transport
- HTTP compatible
- multi-language bindings, IDL

**UDP notifications direct from linecards**
- lightweight, high-scale for interface data
- no reliability, security, IDL, ...
- limited capability on linecard CPUs
Why gRPC for device streaming telemetry

- **leverage HTTP/2** -- persistence, reliability, bidirectional streaming, binary encoding, mTLS security, ...
- **protocol buffers** -- efficient data encodings
- **multiple language bindings** -- client libraries in 10 languages
- **open implementation** -- CNCF project, active community
- **high performance** -- published benchmarks, focus on performance optimizations
- **extensive documentation** -- tutorials, examples, etc. to help vendor partners
State of the art: configuration and commands

Complexity in a multi-vendor, multi-platform environment

- ssh + CLI -- least common denominator
- vendor-specific APIs
- legacy string-based protocols

Non-interoperable “standards”

- e.g., IETF NETCONF/RESTCONF
- inconsistent implementations across vendors

can we also leverage gRPC as a common configuration protocol?
gNMI - gRPC Network Management Interface

single service for state retrieval (telemetry) and state modification (configuration)

- interoperable management requires a specific service definition

option (gnmi_service) = "0.2.0";

```protobuf
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 a stream of values of paths within the data tree.
  rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
```
gNMI adoption in the industry

GA or EFT implementations from multiple vendors

**Routing and switching**
- Cisco
- Arista
- Juniper (now Arista)

**Optical transport**
- Nokia
- Ciena

**WiFi (in progress)**
- Mojo
- Mist
- Aruba (now Juniper)
Lessons and observations in vendor adoption

Detailed service specification was critical for interoperable implementations
● much more than a “gRPC service”

Vendor challenges with working with open source and keeping up with versions
● more pronounced in early stages of gRPC performance and robustness

Workarounds for platform-specific challenges and issues
● inability to handle static constructors due to platform hacks
● need to port gRPC to additional languages (e.g., C)

gRPC lack of common networking features / concepts
● reversible support for dial-out telemetry
● marking traffic with QOS/dscp
More network management services with gRPC

Extend vendor adoption of gRPC and gNMI to additional services

**gNOI - gRPC Network Operations Interface**

- collection of microservices, e.g.
  - reboot, ping, traceroute
  - install / rotate TLS certificates on devices
  - file transfer operations
  - routing protocol operations
- proto encoded data
- implementors select which services to provide
- leverage gRPC reflection for service discovery
- initial implementations from several vendors

**gRIBI - gRPC RIB Interface**

- separate gRPC service to support route injection
- works alongside distributed routing protocols
  - writes to RIB like another routing protocol
  - standard inter-protocol resolution
- initial implementations from multiple vendors
gnmitest - automated software testing based on gNMI

Open source, automated test framework to validate gNMI compliance

Vendors can run test suites before delivering device software releases
- test suites specific to platforms and operational use cases (shared privately)

gnmitest framework

- gRPC service executes tests against network devices using gNMI RPCs

- suite.proto
  - specify gNMI-based telemetry and configuration tests

- gnmitest_cli

- Subscribe, Set, Get

- gRPC service executes tests against network devices using gNMI RPCs
Open source ecosystem

**OpenConfig**

network operator-led open source project

define software abstraction for vendor-neutral device management

- **gnmi**
  - gNMI service definition
  - telemetry collector implementation

- **gnoi**
  - gNOI microservice definitions for operational commands

- **gribi**
  - service definition and reference client/server for gRIBI

- **gnmitest**
  - gNMI Test framework, client, and test suite definitions

[github.com/openconfig/](https://github.com/openconfig/)
gRPC has emerged as a common framework for multiple network mgmt use cases

- monitoring, configuration, operational commands, route injection
- vendor-neutral automated testing

Services and open source tools developed as part of the OpenConfig project

Wide adoption among prominent networking vendors and operators
THANK YOU!