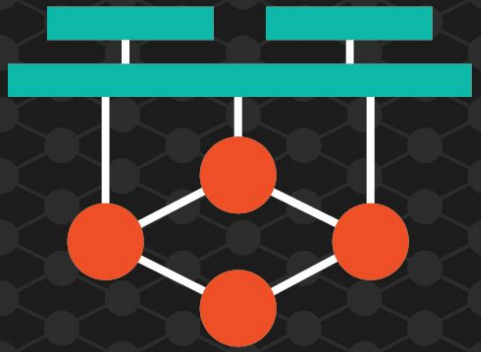
September 25 - 27, 2018
Amsterdam, The Netherlands



ons

EUROPE

OPEN NETWORKING //
Integrate, Automate, Accelerate

ons
EUROPE
OPEN NETWORKING //
Integrate, Automate, Accelerate

September 25 - 27, 2018
Amsterdam, The Netherlands

Programming Network Devices with gRPC and OpenConfig

ons
EUROPE
OPEN NETWORKING //
Integrate, Automate, Accelerate

Nicolas Leiva

- Network Engineer
  - nleiva @ cisco
  - % @
  - % @
  - nleiv4 @

Slides

ons
EUROPE
OPEN NETWORKING //
Integrate, Automate, Accelerate

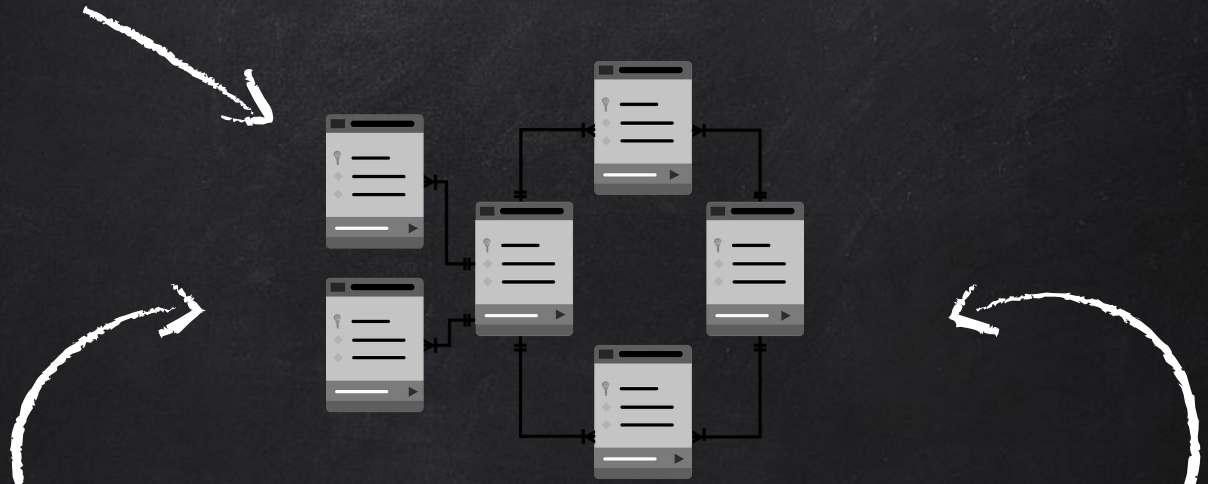
The rise of API's

Beyond the command line

An API is simply a specification of remote calls exposed to the API consumers.

Remote Calls

1. Define the DATA to be transmitted
2. Determine how the data is SERIALIZED over the wire
3. Choose a TRANSPORT protocol



Data Models

Define the data to be transmitted

1

LLDP Neighbors API

Vendor A

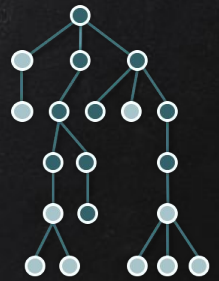
```
{
...
  "TTL": 120,
  "NEIGHBORDEVICE": "ROUTER2.LAB.COM",
  "NEIGHBORPORT": "ETHERNET4",
  "PORT": "ETHERNET4"
...
}
```

Vendor B(J)

```
{
 "LLDP-LOCAL-PORT-ID" [
{
  "DATA" : "ET-0/0/13"
}
...
  "LLDP-REMOTE-PORT-DESCRIPTION" : [
{
   "DATA" : "TO ROUTER3.LAB.COM ET-0/0/13"
}
],
 "LLDP-REMOTE-SYSTEM-NAME" [
{
   "DATA" : "ROUTER3.LAB.COM"
}
```

Vendor C

```
{
..
  "PORT_ID": "ETHERNET2/1",
  "L_PORT_ID": "ETH2/1",
  "SYS_NAME": "ROUTER1.LAB.COM",
  "TTL": 108,
...
}
```



- ✘ Data modeling language
- ✘ Describes data hierarchy
  - config and operational data as a tree structure
- ✘ Specifies restrictions, data types, etc.

ons EUROPE
OPEN NETWORKING //
Integrate, Automate, Accelerate

Interfaces YANG Models (*)

IETF

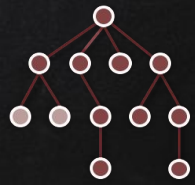
```
module ietf-interfaces {
  revision 2018-02-20 {
    "RFC 8343: ...";  }
  container interfaces {
    list interface {
     leaf name {...}
      leaf type {...}
      leaf enabled {...}
      leaf admin-status {...}
      leaf oper-status {...}
      container statistics {
        leaf in-octets {...}
        leaf out-octets {...}...
```

OpenConfig

```
module openconfig-interfaces {
  revision "2018-04-24" { reference "2.3.1"; }
  container interfaces {
    list interface {
     leaf name {...}
     container config {
      leaf type {...}
      leaf enabled {...}
     container state {
      leaf admin-status {...}
      leaf oper-status {...}
      container counters {
        leaf in-octets {...}
        leaf out-octets {...}
     container subinterfaces {
      list subinterface {
        leaf index {...}...
```

Cisco

```
module Cisco-IOS-XR-ifmgr-cfg {
  revision 2017-09-07 {...}
  container interface-configurations {
    list interface-configuration {
      leaf interface-name {...}
      leaf active {...}
      leaf shutdown {
        type empty;
      }...

module Cisco-IOS-XR-infra-statsd-oper {
  container infra-statistics {
    container interfaces {
      list interface {
        container latest {
          container generic-counters {
            leaf bytes-received {...}
            leaf bytes-sent {...}...
```



OpenConfig Interfaces (*)

```
container interfaces {
  list interface {
      key "name";
      leaf name {...}
      container config {
        uses interface-phys-config;
      }
      container state {
        uses interface-phys-config;
        uses interface-common-state;
        uses interface-counters-state;
      }
      uses interface-phys-holdtime-top;
      uses subinterfaces-top;
  }
}
```

Config

Applied config

Operational State

Statistics

OpenConfig

- ✘ Vendor neutral, driven by network operators
- ✘ Combines config and operational data (intended vs derived state)
  - Config
  - Statistics (e.g., counters)
  - Operational State (e.g., BGP session status)
  - Applied config (...is part of the state)
- ✘ Model consistency and semantic versioning

Encoding

Determine how the data is serialized over the wire

2

Most Common Options

JSON

```
{
  "PERSON": {
    "NAME": "John Doe",
    "EMAIL": "jdoe@example.com"
  }
}
```

XML

```
<PERSON>
  <NAME>John Doe</NAME>
  <EMAIL>jdoe@example.com</EMAIL>
</PERSON>
```

PROTOBUF (*)

```
1 {
  1: "John Doe"
  2: "jdoe@example.com"
}
```

JSON/XML

- ✘ Human readable/editable
- ✘ Can be parsed without knowing schema in advance

```
{
  "person": {
    "name": "John Doe",
    "email": "jdoe@example.com"
  }
}
```

```
<person>
  <name>John Doe</name>
  <email>jdoe@example.com</email>
</person>
```

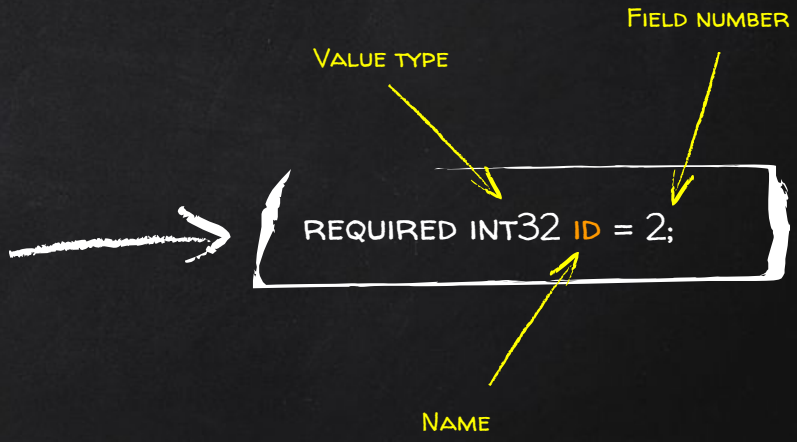
Protocol Buffers

- ✘ Very dense data (small output)
- ✘ Very fast processing
- ✘ Not human readable (native format)
- ✘ Only meaningful if you have the message definition

```
1 {
  1: "John Doe"
  2: "jdoe@example.com"
}
```

Protocol Buffers

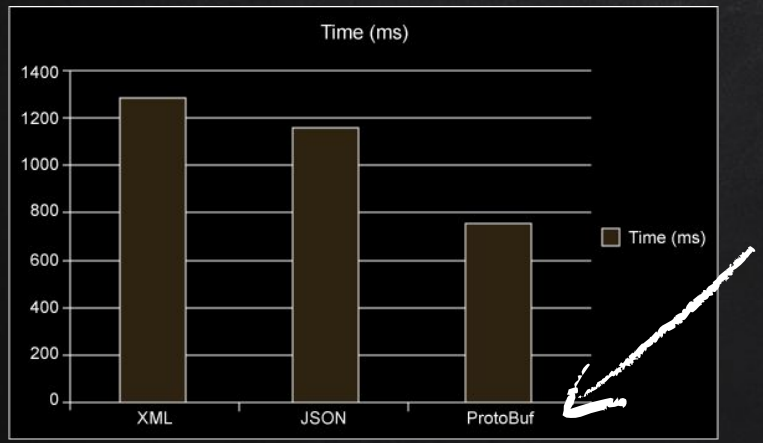
```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }
  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }
  repeated PhoneNumber phone = 4;
}
```



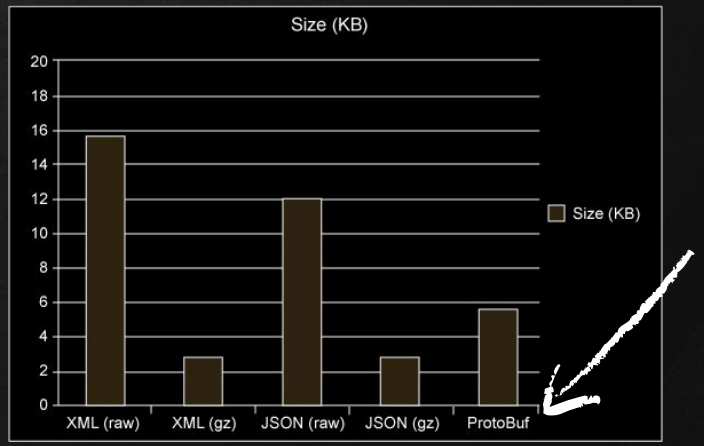
.proto file

Protocol Buffers

Comparing data-format speeds



Size of data by format



Transport

Choose a transport protocol

3

Transport mechanisms

NETCONF

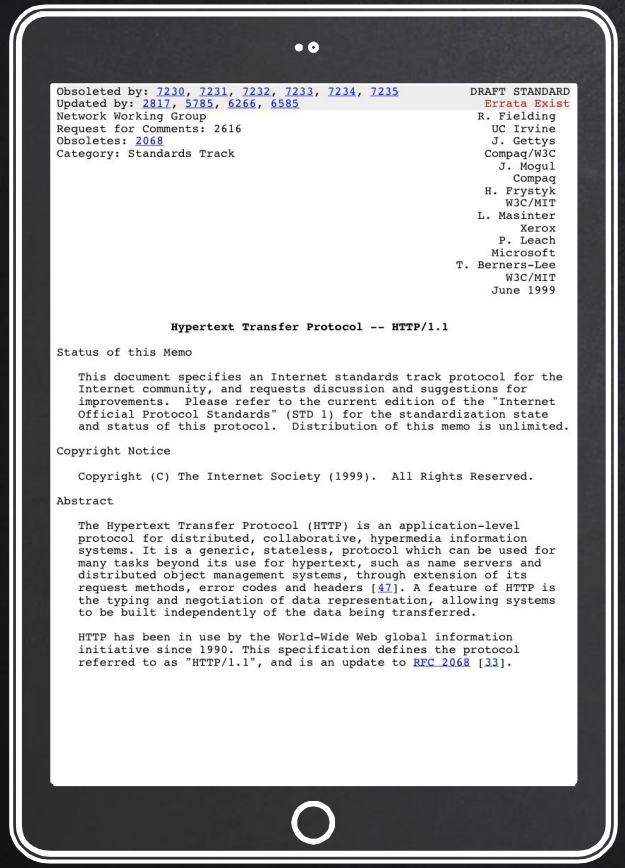
- ✘ SSH
- ✘ RPC
  - ○ <get-config>
  - ○ <edit-config>
  - ○ <commit>
  - ○ <lock>
  - ○ ...

RESTCONF

- ✘ HTTP
- ✘ Methods
  - ○ GET
  - ○ POST
  - ○ DELETE
  - ○ PUT
  - ○ ...

gRPC

- ✘ HTTP/2
- ✘ RPC
  - ○ Unary
  - ○ Server streaming
  - ○ Client streaming
  - ○ Bidirectional streaming



Obsoleted by: 7230, 7231, 7232, 7233, 7234, 7235 DRAFT STANDARD
Updated by: 2817, 5785, 6266, 6585 Errata Exist
Network Working Group R. Fielding
Request for Comments: 2616 UC Irvine
Obsoletes: 2068 J. Gettys
Category: Standards Track Compaq/W3C
J. Mogul
Compaq
H. Frystyk
W3C/MIT
L. Masinter
Xerox
P. Leach
Microsoft
T. Berners-Lee
W3C/MIT
June 1999

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [47]. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1", and is an update to RFC 2068 [33].

HTTP/1.1

Jun 1999

Network Working Group T. Ylonen
Request for Comments: 4253 SSH Communications Security Corp
Category: Standards Track C. Lonvick, Ed.
Cisco Systems, Inc.
January 2006

The Secure Shell (SSH) Transport Layer Protocol

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

The Secure Shell (SSH) is a protocol for secure remote login and other secure network services over an insecure network.

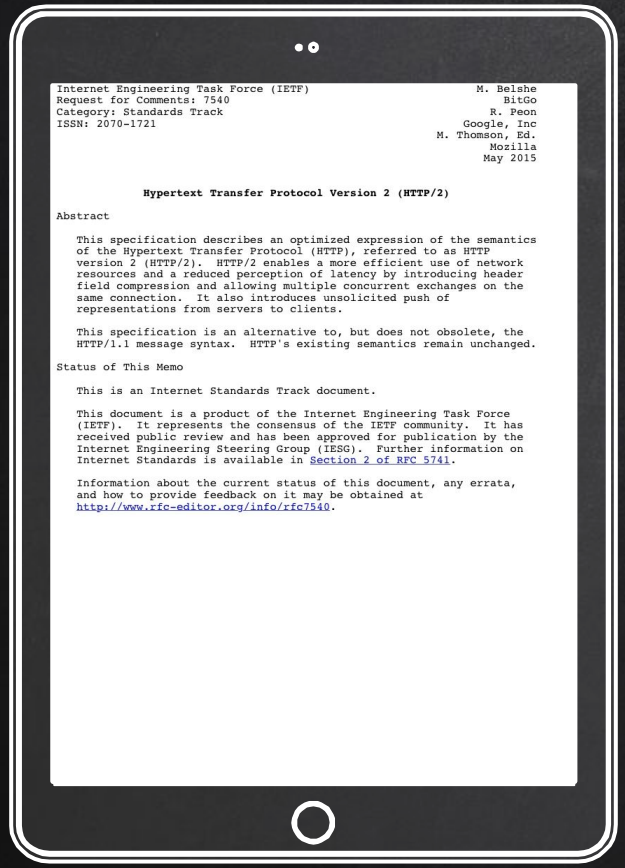
This document describes the SSH transport layer protocol, which typically runs on top of TCP/IP. The protocol can be used as a basis for a number of secure network services. It provides strong encryption, server authentication, and integrity protection. It may also provide compression.

Key exchange method, public key algorithm, symmetric encryption algorithm, message authentication algorithm, and hash algorithm are all negotiated.

This document also describes the Diffie-Hellman key exchange method and the minimal set of algorithms that are needed to implement the SSH transport layer protocol.

SSH-2

Jan 2006



Internet Engineering Task Force (IETF) M. Belshe
Request for Comments: 7540 BitGo
Category: Standards Track R. Peon
ISSN: 2070-1721 Google, Inc
M. Thomson, Ed.
Mozilla
May 2015

Hypertext Transfer Protocol Version 2 (HTTP/2)

Abstract

This specification describes an optimized expression of the semantics of the Hypertext Transfer Protocol (HTTP), referred to as HTTP version 2 (HTTP/2). HTTP/2 enables a more efficient use of network resources and a reduced perception of latency by introducing header field compression and allowing multiple concurrent exchanges on the same connection. It also introduces unsolicited push of representations from servers to clients.

This specification is an alternative to, but does not obsolete, the HTTP/1.1 message syntax. HTTP's existing semantics remain unchanged.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7540.

HTTP/2

May 2015

HTTP/2

- ✘ Binary, easier framing
- ✘ Header compression
- ✘ request and response multiplexing over a single TCP connection
- ✘ bidirectional streams

gRPC

- ✘ Strongly typed service and message definition
- ✘ Takes care of all the underlying plumbing
- ✘ Runs over HTTP/2
- ✘ Cloud Native Computing Foundation Project

gRPC

send 300,000 requests to key/value store

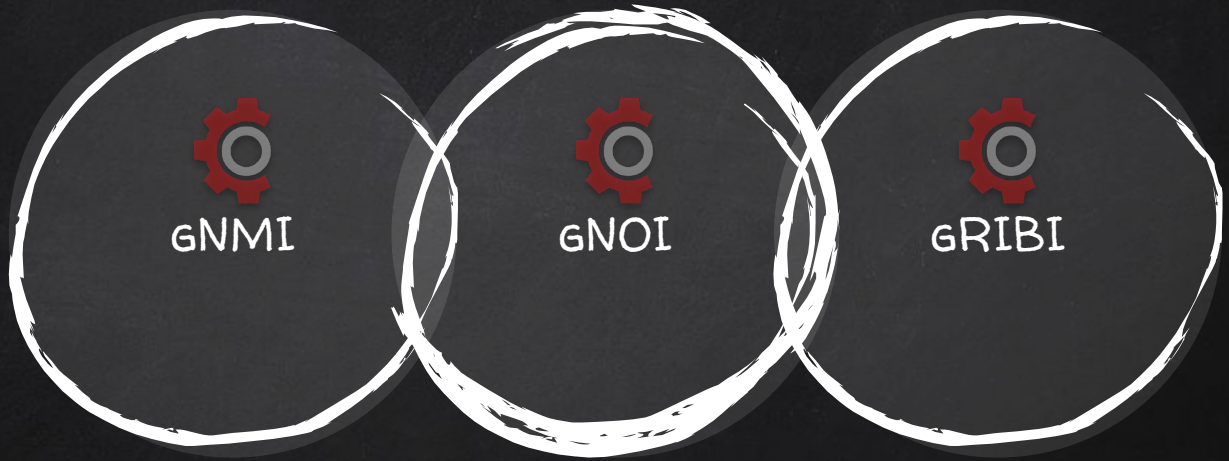
Processing time

RPC	# of clients	total time	per-request time
jsonrpc	1	8m 7.2s	1.624ms
gRPC	1	36.7s	122.3µs
gRPC	100	7.1s	23.8µs

memory usage

RPC	# of clients	AllocsPerOp	AllocedBytesPerOp
jsonrpc	1	32.7M	3.1GB
gRPC	1	25.2M	1.7GB
gRPC	100	25.2M	1.7GB

OpenConfig gRPC Interfaces



gRPC service interface definitions

gNMI

```
service gNMI {
  rpc Capabilities(CapabilityRequest)
returns (CapabilityResponse);
  rpc Get(GetRequest) returns
(GetResponse);
  rpc Set(SetRequest) returns
(SetResponse);
  rpc Subscribe(stream SubscribeRequest)
returns (stream SubscribeResponse);
}
```

gNOI

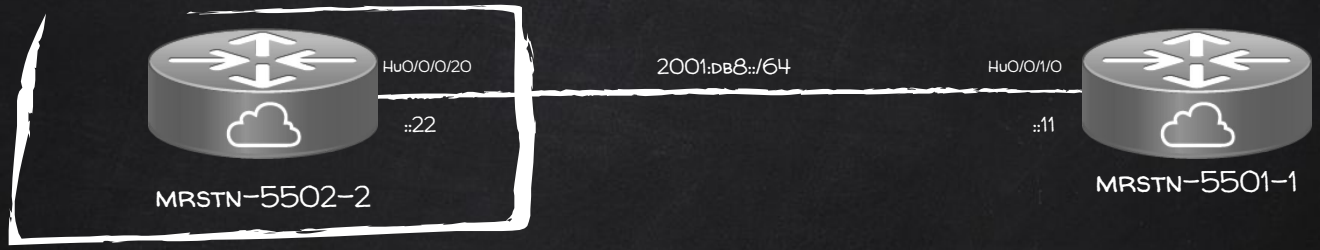
```
service System {
  rpc Ping(PingRequest) returns (stream
PingResponse) {}
  rpc Traceroute(TracerouteRequest)
returns (stream TracerouteResponse) {}
  rpc Time(TimeRequest) returns
(TimeResponse) {}
  rpc SetPackage(stream
SetPackageRequest) returns
(SetPackageResponse) {}
  ...
}
```

Cisco

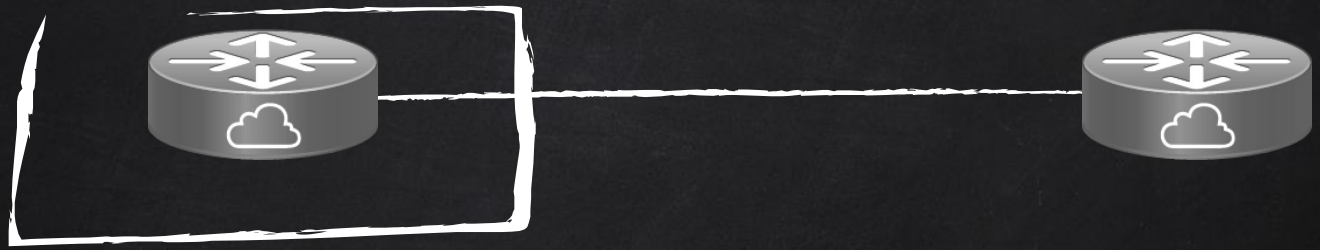
```
service gRPCConfigOper {
    rpc GetConfig(ConfigGetArgs)
returns(stream ConfigGetReply) {};
    rpc MergeConfig(ConfigArgs)
returns(ConfigReply) {};
...
    rpc CreateSubs(CreateSubsArgs)
returns(stream CreateSubsReply) {};
}

service gRPCExec {
...
    rpc ActionJSON(ActionJSONArgs)
returns(stream ActionJSONReply) {};
}
```

Demo Time

Any questions?


Hu0/0/0/20
::22
2001:DB8::/64
Hu0/0/1/0
::11
MRSTN-5502-2
MRSTN-5501-1


CONFIG



Is the session still UP?

Am I receiving all the prefixes?

Telemetry

Prefix delta > 10–30%?

Am I receiving more prefixes than expected?

ons
EUROPE
OPEN NETWORKING //
Integrate, Automate, Accelerate

```
grouping bgp-neighbor-state {
  description
   "Operational state parameters relating only to a BGP neighbor";

  leaf session-state {
   type enumeration {
      ...
    }
   description
    "Operational state of the BGP peer";
  }
```

```
enum IDLE {
        description
          "neighbor is down, and in the Idle state of the
          FSM";
}
enum CONNECT {
        description
          "neighbor is down, and the session is waiting for
          the underlying transport session to be established";
}
enum ACTIVE {
        description
          "neighbor is down, and the local system is awaiting
          a conncetion from the remote peer";
}
enum OPENSENT {
        description
          "neighbor is in the process of being established.
          The local system has sent an OPEN message";
}
enum OPENCONFIRM {
        description
          "neighbor is in the process of being established.
          The local system is awaiting a NOTIFICATION or
          KEEPALIVE message";
}
enum ESTABLISHED {
        description
          "neighbor is up - the BGP session with the peer is
          established";
}
```

https://github.com/nleiva/xroc

Further reading

gRPC and GPB for Networking Engineers

https://github.com/nleiva/gmessaging

Programming IOS-XR with gRPC and Go

https://xrdocs.github.io/programmability/tutorials/2017-08-04-programming-ios-xr-with-grpc-and-go/

Validate the intent of network config changes

https://xrdocs.github.io/programmability/tutorials/2017-08-14-validate-the-intent-of-network-config-changes/

ygot (YANG Go Tools)

https://github.com/openconfig/ygot

YANG Development Kit (YDK)

https://developer.cisco.com/site/ydk/

OpenConfig GitHub

https://github.com/openconfig