cisco Live!

Let’s go

The bridge to possible

Streaming Telemetry on Cisco NX-OS

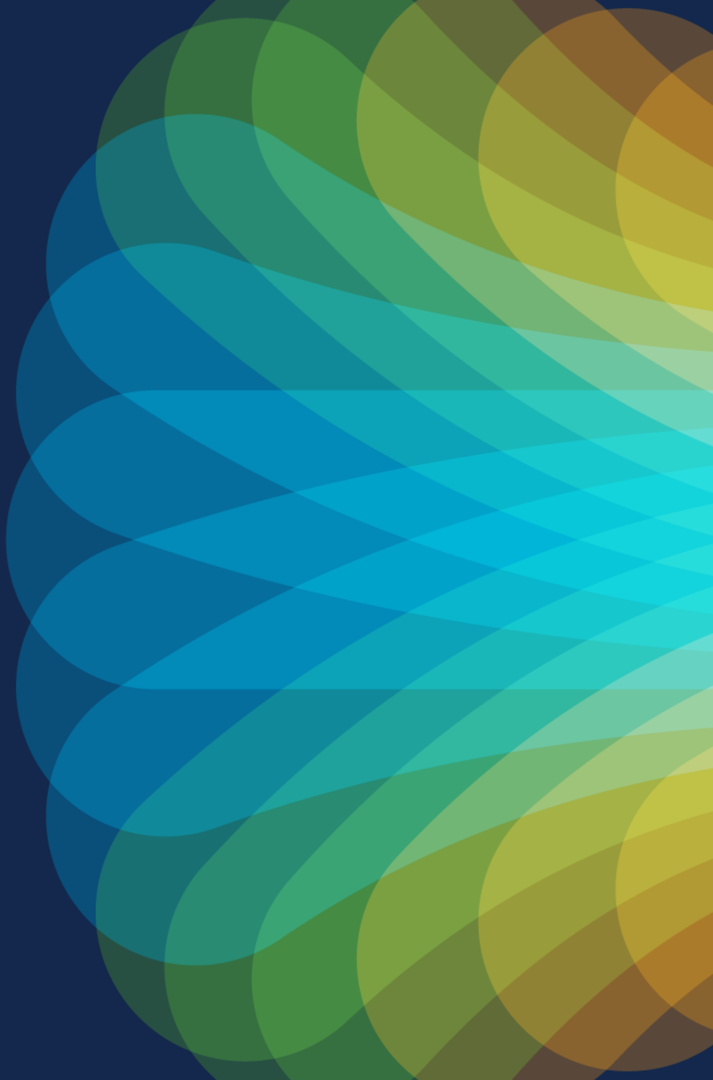
Nick Mortari
Technical Marketing Engineer
Cloud Networking Team

BRKDCN-2689

Agenda

- Why New Telemetry Methods?
- Building Blocks of Streaming Telemetry on NX-OS
- How to Create an Open-Source Telemetry System
- Live Demo

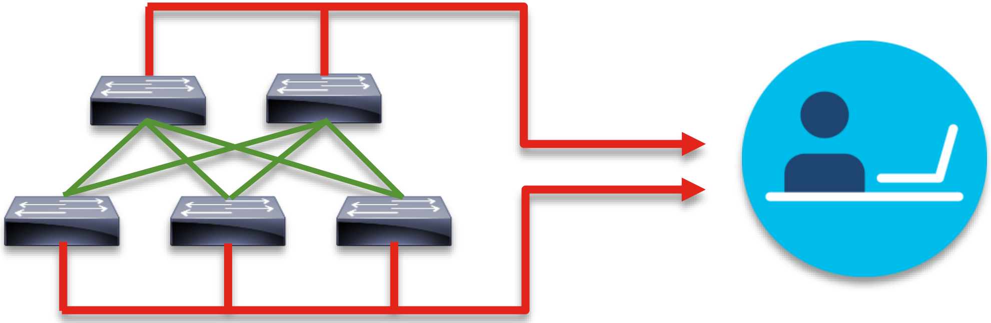
What Is Telemetry?



What Is Telemetry?

Real-time collection of device information

Example: What is the current power consumption of my switches?



Why New Telemetry Methods?

Why New Telemetry Methods?

I can't get the information I need...

My collection methods don't scale well...

It's hard to process the data I receive...

I want to compare data with other devices...

I can't poll often enough...

Why New Telemetry Methods?

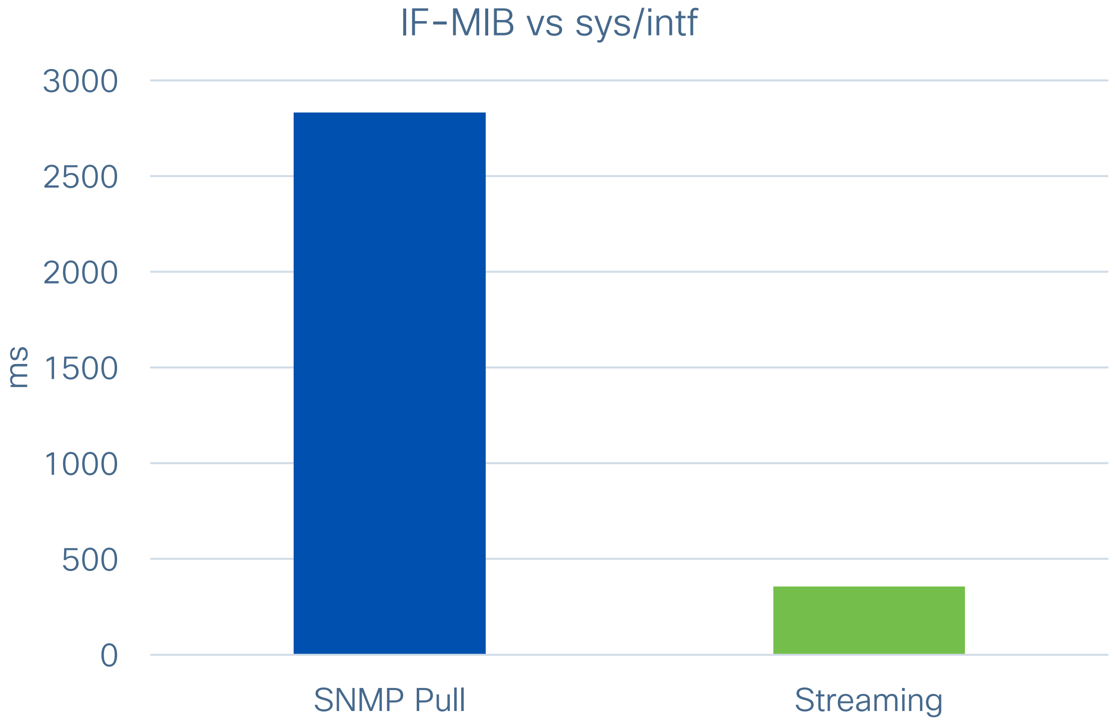
Detailed Information

Scalability

Performance

Why New Telemetry Methods?

Performance



🟦 SNMP bulk-get `IF-MIB`

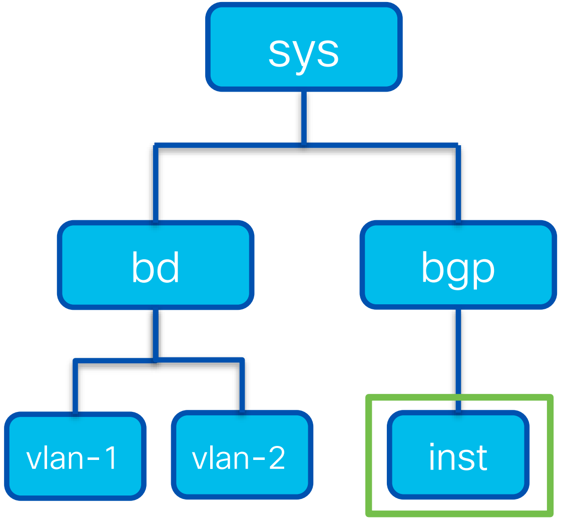
🟩 Streaming DME sensor path `sys/intf`

Building Blocks of Streaming Telemetry

- **Data Sources**
- Data Frequency
- Data Encoding
- Data Transport

Data Sources

DME(Data Management Engine)

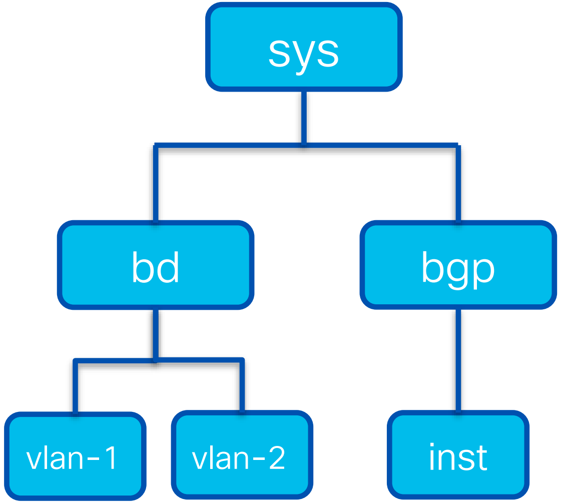


- Configuration and operational data is stored in DME
- Tree data structure
- DN (Distinguished Name) is in .../.../.../... format
- Telemetry data can be accessed with the DN as a sensor path

- sys/bgp/inst represents configuration and state data for BGP process

Data Sources

What Is Available in DME?

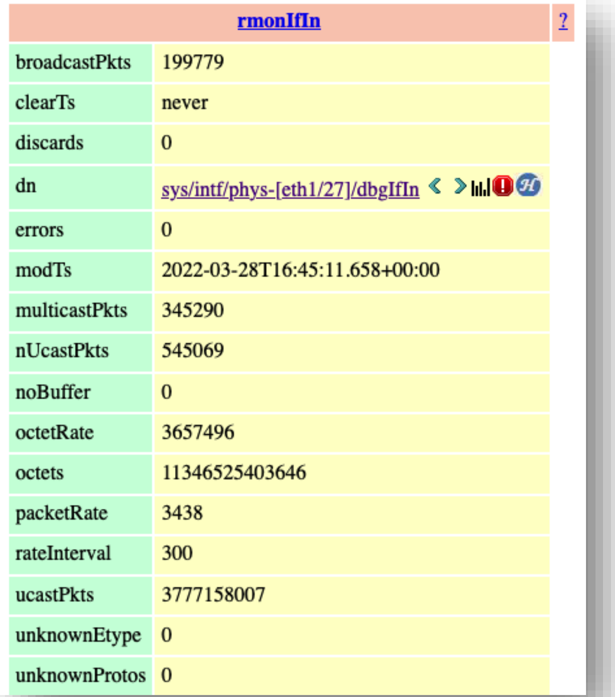


- Almost entire OS is available
- As of 10.4(2)F, over 95% of the commands are DMEized
- Supports event-based and sample-based telemetry
- Extra filters are supported to minimize data size

Data Sources

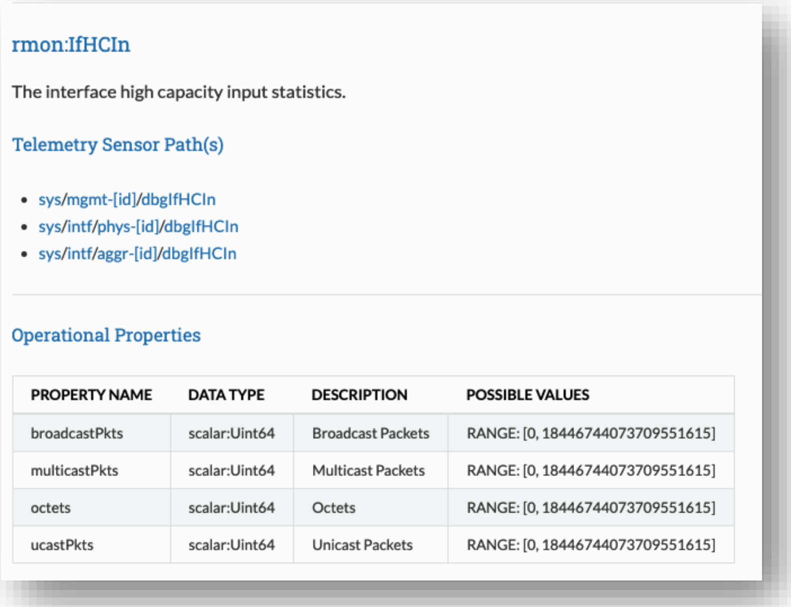
How to Get Sensor Paths for DME

Visore is a built-in DME browser of NX-OS, navigate to https://[ip_of_swtich]/visore.html



rmonIfIn	?
broadcastPkts	199779
clearTs	never
discards	0
dn	sys/intf/phys-[eth1/27]/dbgIfIn
errors	0
modTs	2022-03-28T16:45:11.658+00:00
multicastPkts	345290
nUcastPkts	545069
noBuffer	0
octetRate	3657496
octets	11346525403646
packetRate	3438
rateInterval	300
ucastPkts	3777158007
unknownEtype	0
unknownProtos	0

API reference is also available: https://developer.cisco.com/site/nxapi-dme-model-reference-api/?version=10.2(2)



rmon:IfHCIn

The interface high capacity input statistics.

Telemetry Sensor Path(s)

- sys/mgmt-[id]/dbgIfHCIn
- sys/intf/phys-[id]/dbgIfHCIn
- sys/intf/aggr-[id]/dbgIfHCIn

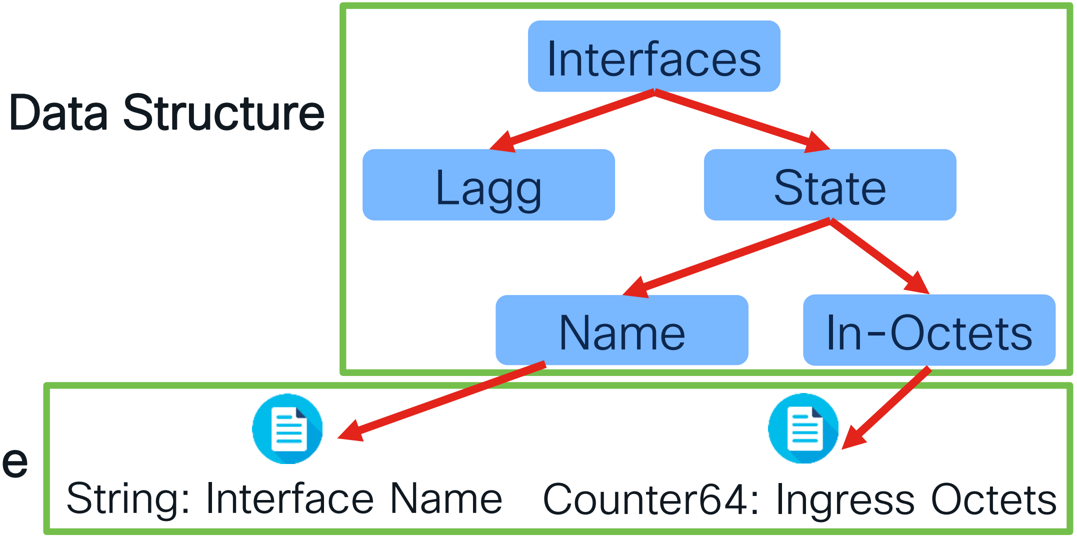
Operational Properties

PROPERTY NAME	DATA TYPE	DESCRIPTION	POSSIBLE VALUES
broadcastPkts	scalar:Uint64	Broadcast Packets	RANGE: [0, 18446744073709551615]
multicastPkts	scalar:Uint64	Multicast Packets	RANGE: [0, 18446744073709551615]
octets	scalar:Uint64	Octets	RANGE: [0, 18446744073709551615]
ucastPkts	scalar:Uint64	Unicast Packets	RANGE: [0, 18446744073709551615]

Data Sources

YANG Models

- YANG (Yet Another Next Generation) is a data modeling language
- Defines the data structure and data type for the model we use



Data Sources

YANG Models

- NX-OS supports two YANG models for telemetry
  - OpenConfig YANG model
  - Cisco native model
- You can access telemetry from each model with an XPATH (XML Path)

Example of xpath:

```
openconfig-interfaces:interfaces/interface/state/oper-status
```

Model name

Container

List

Leaf

Data Sources

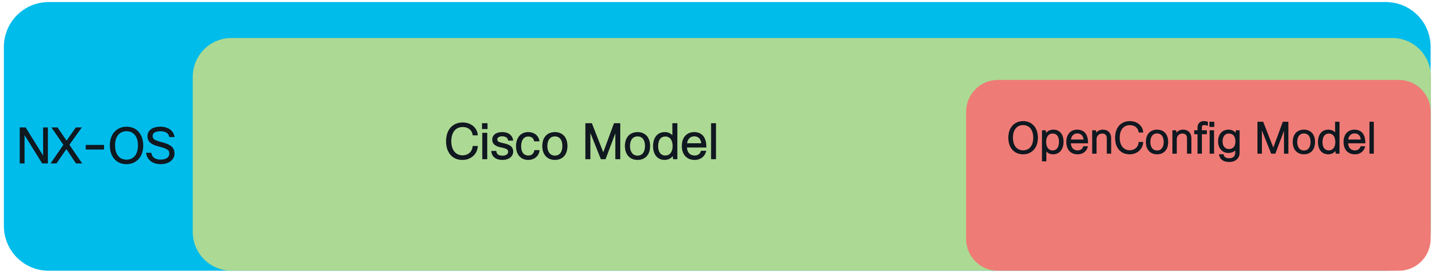
YANG Models

Cisco Native Model

- Vendor specific
- Created by Cisco
- Supports almost every feature on NX-OS

OpenConfig Model

- Vendor agnostic
- Created by many networking companies (open-source)
- Does not support every feature on NX-OS



Data Sources

Supported OC YANG Modules

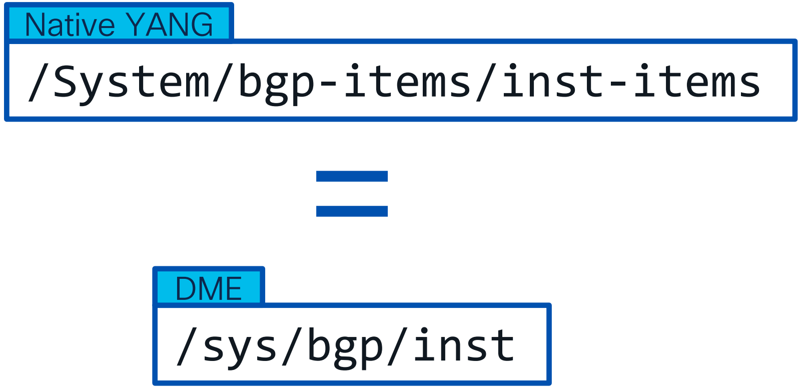
model	Revision in 10.3(3)F
openconfig-aaa.yang	2019-10-28
openconfig-acl.yang	2019-11-27
openconfig-bfd.yang	2020-05-08
openconfig-bgp.yang	2019-07-10
openconfig-igmp.yang	2019-07-09
openconfig-interfaces.yang	2019-11-19
openconfig-isis.yang	2020-03-24
openconfig-lacp.yang	2018-11-21
openconfig-lldp.yang	2018-11-21
openconfig-mpls.yang	2019-03-26
openconfig-network-instance.yang	2022-04-20
openconfig-ospfv2.yang	2019-11-28
openconfig-pim.yang	2019-07-09
openconfig-platform.yang	2019-04-16
openconfig-qos.yang	2019-11-28
openconfig-routing-policy.yang	2018-11-21
openconfig-system.yang	2020-03-25

- To support OC YANG
  - Before 10.2(2)F, `mtx-openconfig-all` rpm needs to be installed on the streaming switch
  - After 10.2(2)F, use `feature openconfig` to enable
- Beware of deviations, it is possible to partially support a module
  - A deviation is when a path is not following the definition in OC module, or the path is not supported

- A full list of supported modules and deviations is published on GitHub: https://github.com/YangModels/yang/tree/master/vendor/cisco/nx

Data Sources

Native YANG



Native YANG

`/System/bgp-items/inst-items`

=

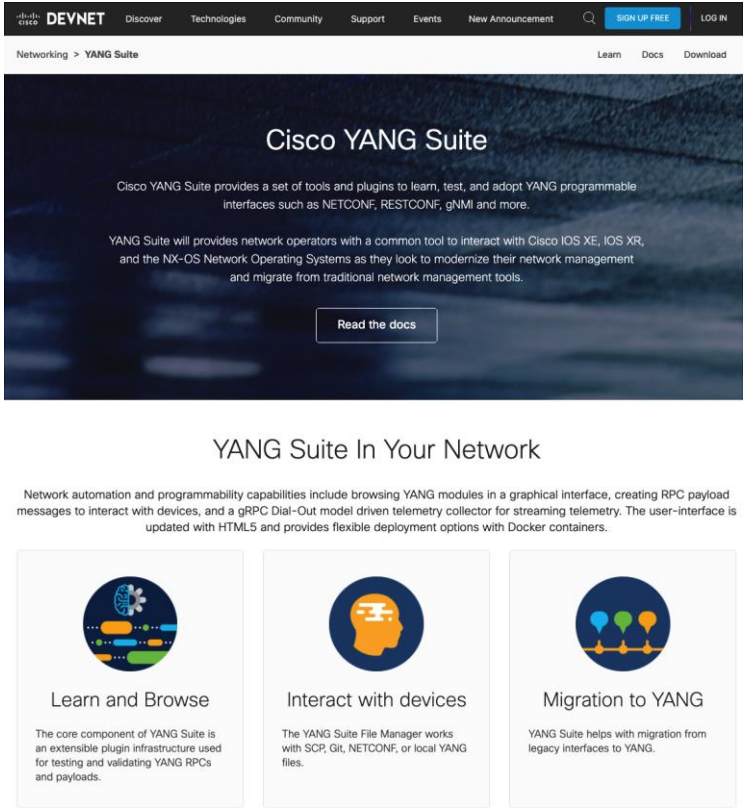
DME

`/sys/bgp/inst`

- NX-OS Native YANG is defined in the *Cisco-NX-OS-device.yang module*
- It is a 1:1 mapping from DME objects to Native YANG

YANG Suite

The Swiss Army Knife of YANG



Cisco YANG Suite

Cisco YANG Suite provides a set of tools and plugins to learn, test, and adopt YANG programmable interfaces such as NETCONF, RESTCONF, gNMI and more.

YANG Suite will provides network operators with a common tool to interact with Cisco IOS XE, IOS XR, and the NX-OS Network Operating Systems as they look to modernize their network management and migrate from traditional network management tools.

YANG Suite In Your Network

Network automation and programmability capabilities include browsing YANG modules in a graphical interface, creating RPC payload messages to interact with devices, and a gRPC Dial-Out model driven telemetry collector for streaming telemetry. The user-interface is updated with HTML5 and provides flexible deployment options with Docker containers.

Learn and Browse

The core component of YANG Suite is an extensible plugin infrastructure used for testing and validating YANG RPCs and payloads.

Interact with devices

The YANG Suite File Manager works with SCP, Git, NETCONF, or local YANG files.

Migration to YANG

YANG Suite helps with migration from legacy interfaces to YANG.

- One-stop tool for automating network devices using the YANG model
- Construct and test YANG based API interface over NETCONF, RESTCONF and gNMI
- YANG model browser built-in

https://developer.cisco.com/yangsuite

Data Sources

CLI/NX-API

```
93240YC-FX2-L02-S4# show nve vni  | json-pretty
{
    "TABLE_nve_vni": {
        "ROW_nve_vni": [
            {
                "if-name": "nve1",
                "vni": "30000",
                "mcast": "239.1.1.1",
                "vni-state": "Up",
                "mode": "CP",
                "type": "L2 [2300]",
                "flags": null,
                "dci-mcast": "Unconfigured"
            },
...
```

- 100% of customer-facing show commands of NX-OS have structured output

- Only supports sample-based telemetry
- CLI doesn‘t have data types, all values are strings
    - The collector will need to parse the result and “guess” data type

Platform Support for Data Sources

Nexus Platform	DME	CLI/NX-API	YANG	Release
3000 with 8G+ RAM	✓	✓	✓ *	7.0(3)I7(1)
9300	✓	✓	✓ *	7.0(3)I5(1)
9500/9400/9800	✓	✓	✓ *	7.0(3)I7(1)
7000/7700	✗	✓	✗	8.3(1)

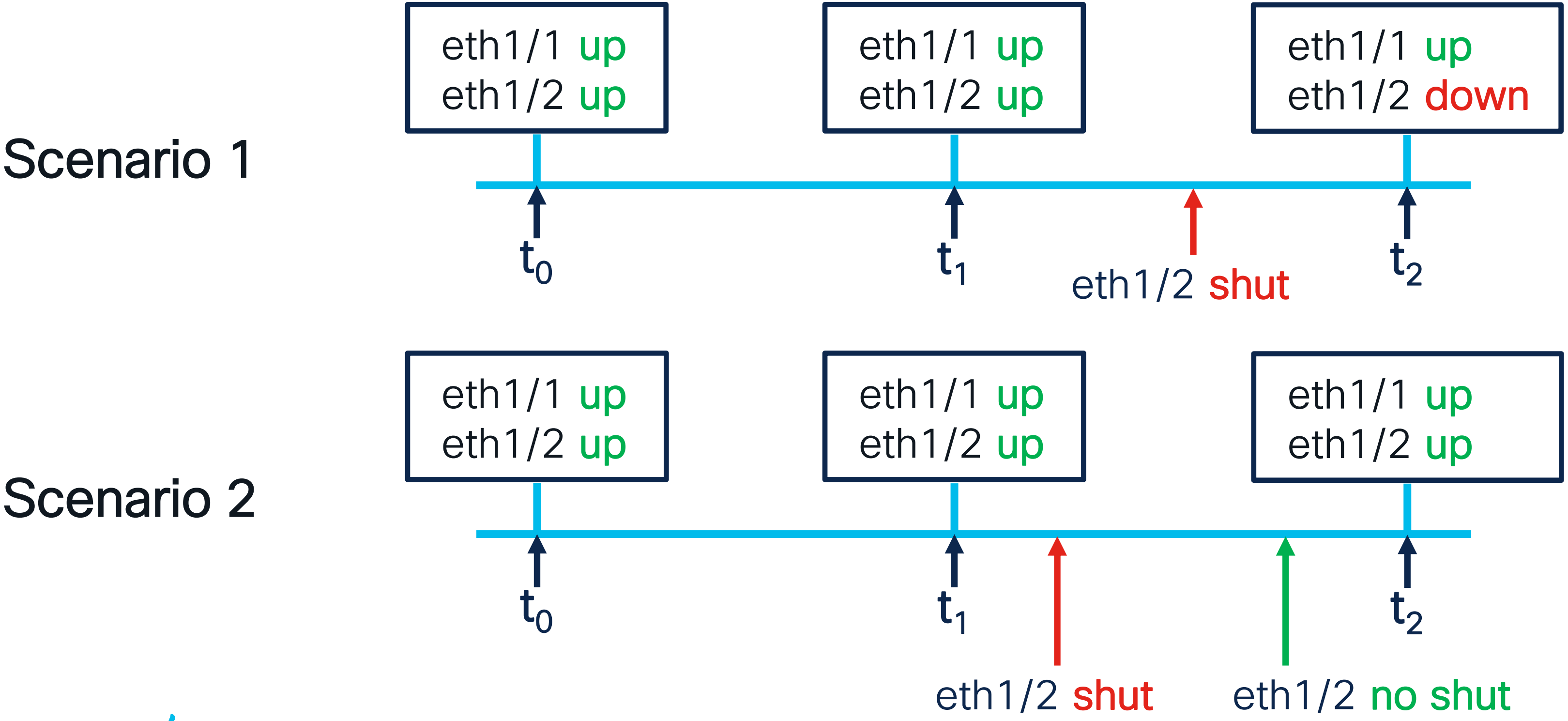
* Streaming YANG models starting from 9.2(1)

Building Blocks of Streaming Telemetry

- Data Sources
- **Data Frequency**
- Data Encoding
- Data Transport

Data Frequency

Sample-Based Telemetry



Scenario 1

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 down

t_0 t_1 eth1/2 shut t_2

Scenario 2

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 up

eth1/1 up
eth1/2 up

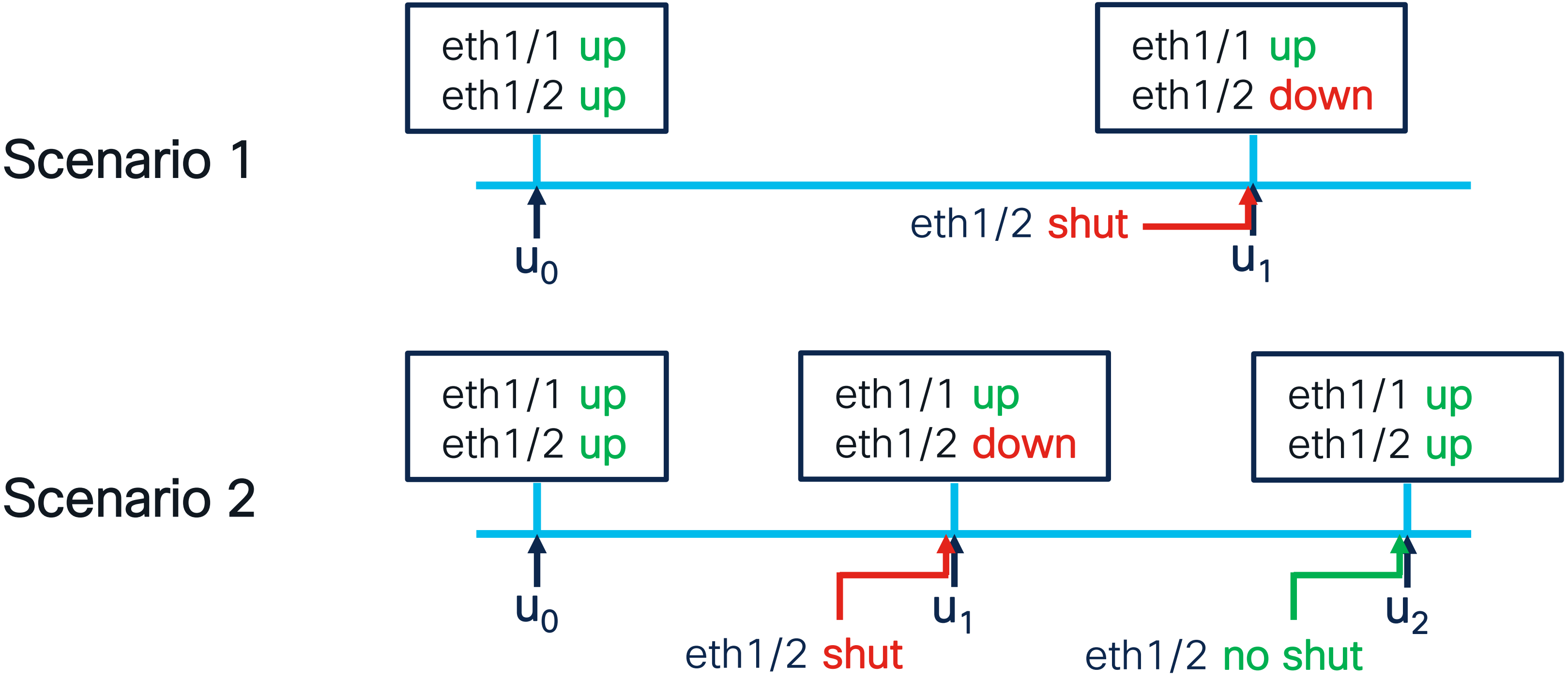
t_0 t_1 t_2

eth1/2 shut

eth1/2 no shut

Data Frequency

Event-Based Telemetry

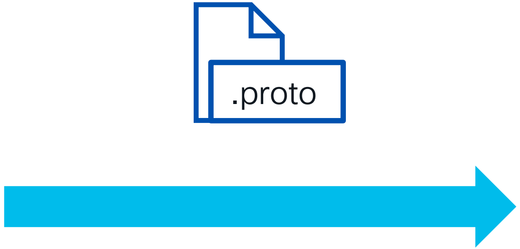


Building Blocks of Streaming Telemetry

- Data Sources
- Data Frequency
- **Data Encoding**
- Data Transport

How Does GPB(Google Protocol Buffers) Work?

```
<interface>
  <name>eth1/49</name>
  <state>
    <counters>
      <in-broadcast-pkts>2</in-broadcast-pkts>
      <in-discards>0</in-discards>
      <in-errors>0</in-errors>
      <in-fcs-errors>0</in-fcs-errors>
      <in-multicast-pkts>30543</in-multicast-pkts>
      <in-octets>13320913920</in-octets>
      <in-unicast-pkts>5406026</in-unicast-pkts>
      <in-unknown-protos>0</in-unknown-protos>
      <out-broadcast-pkts>3</out-broadcast-pkts>
      <out-discards>0</out-discards>
      <out-errors>0</out-errors>
      <out-multicast-pkts>26070</out-multicast-pkts>
      <out-octets>143144868</out-octets>
      <out-unicast-pkts>1424051</out-unicast-pkts>
    </counters>
  </state>
</interface>
```



```
1:"eth1/49"
2:{
  1:{
    1:2
    2:0
    3:0
    4:0
    5:30543
    6:13320913920
    7:5406026
    8:0
    9:3
    10:0
    11:0
    12:26070
    13:143144868
    14:1424051
  }
}
```

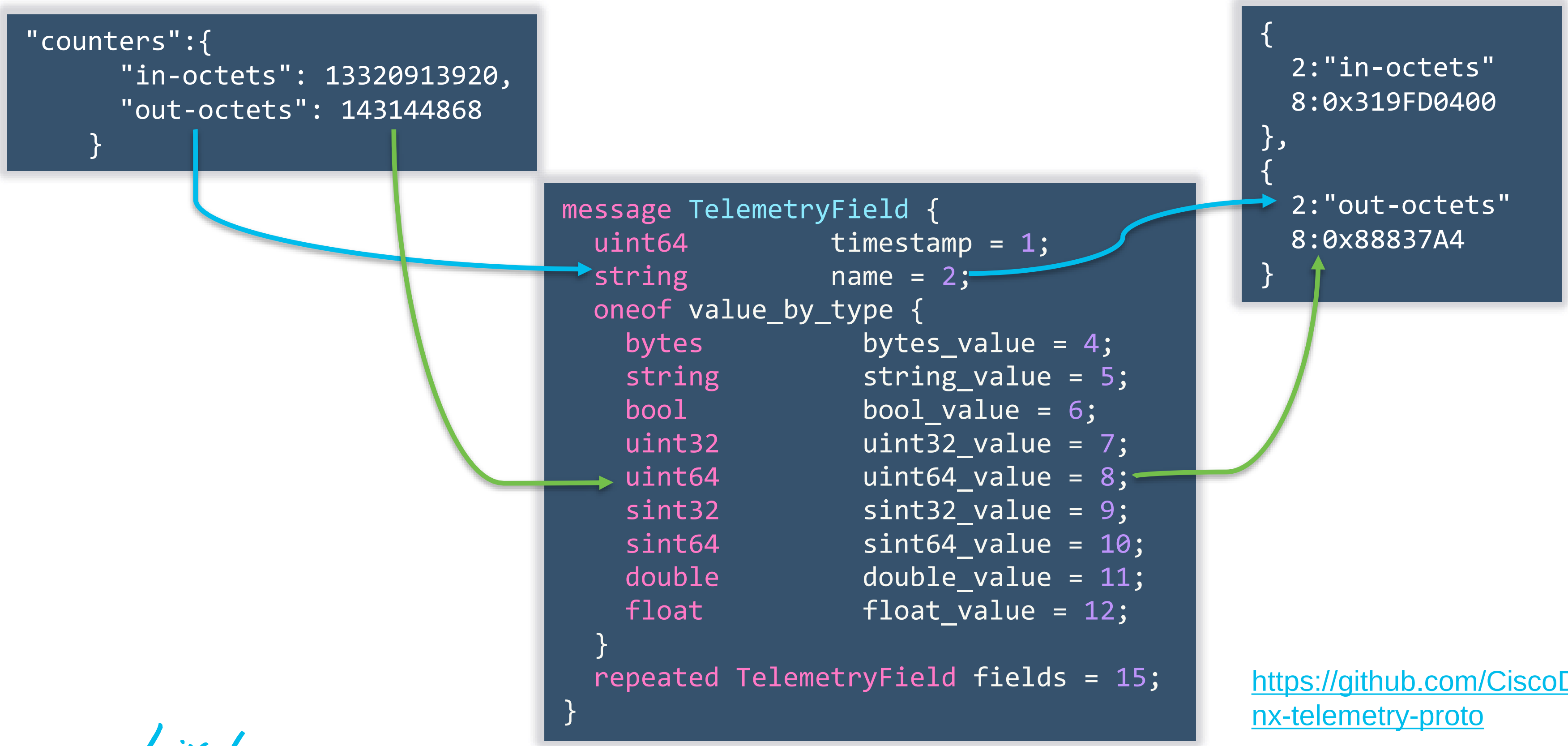
How Does GPB(Google Protocol Buffers) Work?

```
<interface>
  <name>eth1/49</name>
  <state>
    <counters>
      <in-broadcast-pkts>2</in-broadcast-pkts>
      <in-discards>0</in-discards>
      <in-errors>0</in-errors>
      <in-fcs-errors>0</in-fcs-errors>
      <in-multicast-pkts>30543</in-multicast-pkts>
      <in-octets>13320913920</in-octets>
      <in-unicast-pkts>5406026</in-unicast-pkts>
      <in-unknown-protos>0</in-unknown-protos>
      <out-broadcast-pkts>3</out-broadcast-pkts>
      <out-discards>0</out-discards>
      <out-errors>0</out-errors>
      <out-multicast-pkts>26070</out-multicast-pkts>
      <out-octets>143144868</out-oc
      <out-unicast-pkts>1424051</ou
    </counters>
  </state>
</interface>
```

```
1:"eth1/49"
2:{
  1:{
    1:2
    2:0
    3:0
    4:0
    5:30543
    6:13320913920
    7:5406026
    8:0
    9:3
    10:0
    11:0
    12:26070
          43144868
          424051
```

High wire efficiency,
but hard to develop the encoder and decoder

How Does GPB-KV (Key-Value) Work?



```
"counters":{
      "in-octets": 13320913920,
      "out-octets": 143144868
    }
```

```
message TelemetryField {
  uint64          timestamp = 1;
  string          name = 2;
  oneof value_by_type {
    bytes             bytes_value = 4;
    string            string_value = 5;
    bool              bool_value = 6;
    uint32            uint32_value = 7;
    uint64            uint64_value = 8;
    sint32            sint32_value = 9;
    sint64            sint64_value = 10;
    double            double_value = 11;
    float             float_value = 12;
  }
  repeated TelemetryField fields = 15;
}
```

```
{
  2:"in-octets"
  8:0x319FD0400
},
{
  2:"out-octets"
  8:0x88837A4
}
```

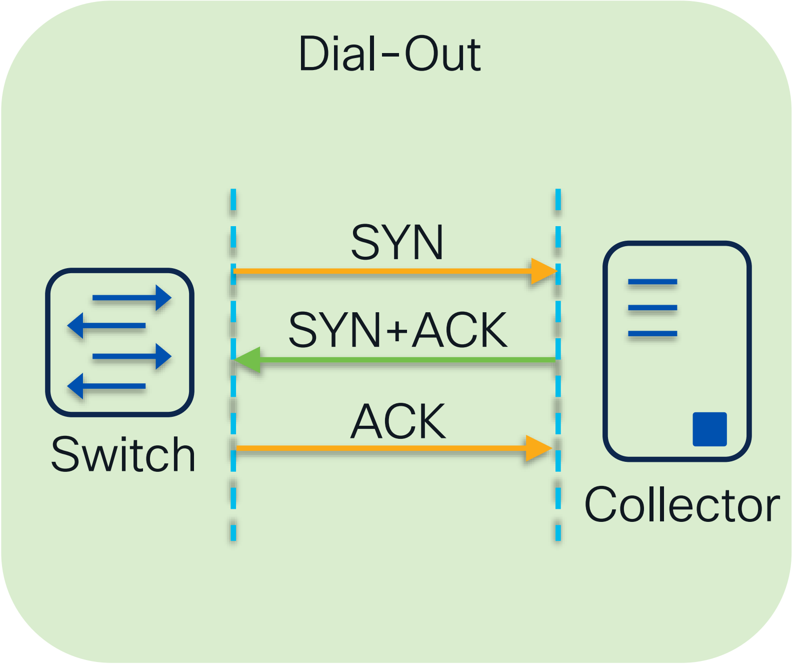
https://github.com/CiscoDevNet/nx-telemetry-proto

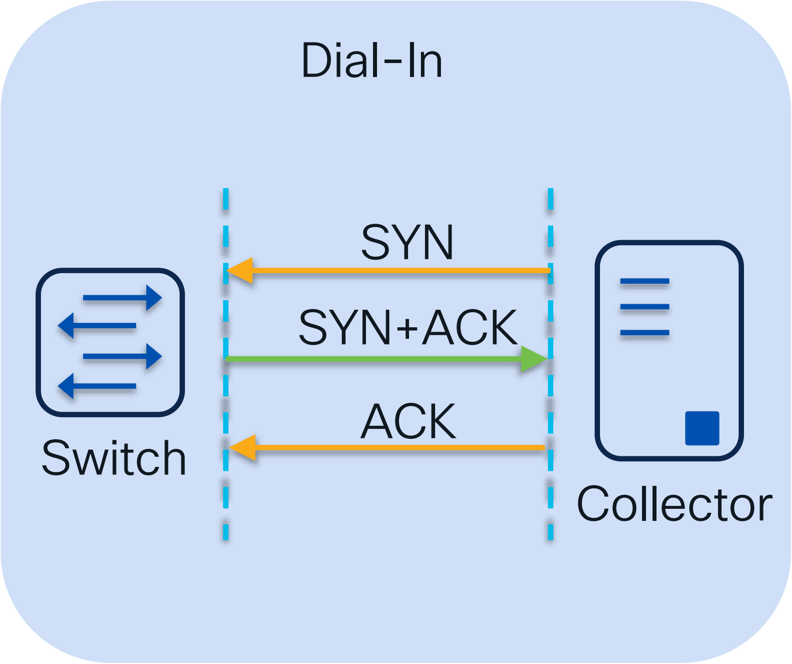
Building Blocks of Streaming Telemetry

- Data Sources
- Data Frequency
- Data Encoding
- **Data Transport**

Dial-Out vs Dial-In

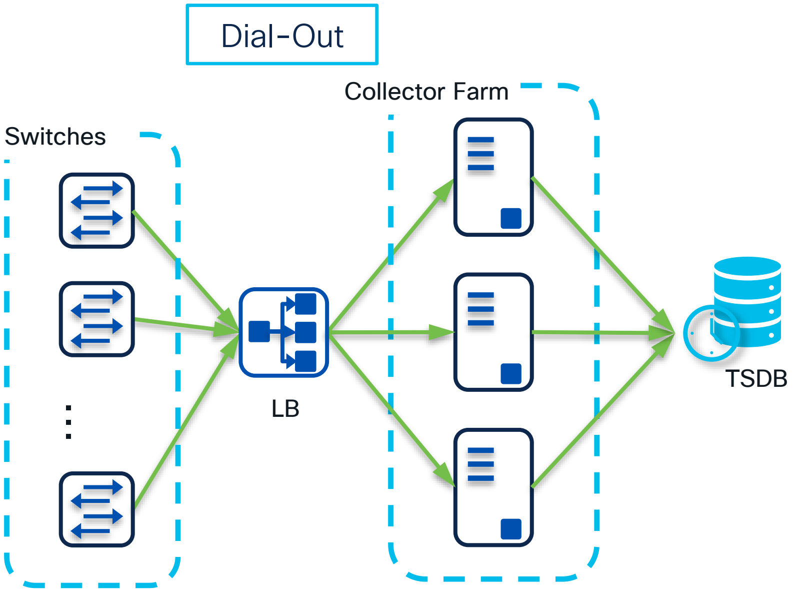
- TCP connection is always persistent in telemetry
- The difference is which part initializes the connection



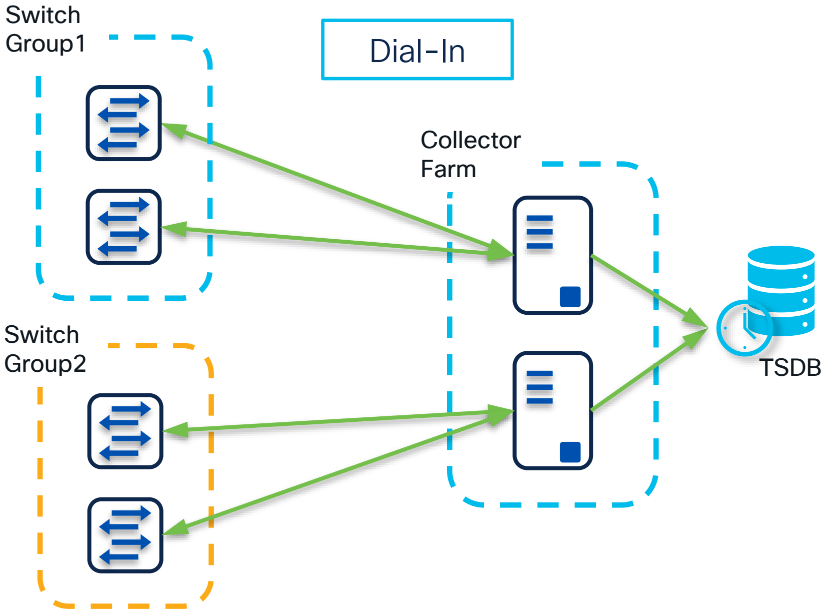


Dial-Out vs Dial-In

Design Considerations



Collectors can be set up behind load balancer, all switches stream to the same VIP of collector



To distribute the workload, the collectors need to dial-in to different switch groups, need to keep the sensor configuration synchronized across the cluster

Dial-Out vs Dial-In

Dial-Out	Dial-In
Support gRPC, HTTP, UDP as the transport protocol	Only gNMI is supported as the protocol
Configuration needs to be done from CLI or other management interfaces	Single channel for subscription and data transport
No need to open a specific port to the management interface of the switch	External firewall must allow ingress connection to gNMI service on switch
Load balancing is easier by setting up collector behind VIP	gNMI clients need to be distributed across switches

gNMI

gRPC Network Management Interface

gNMI Introduction

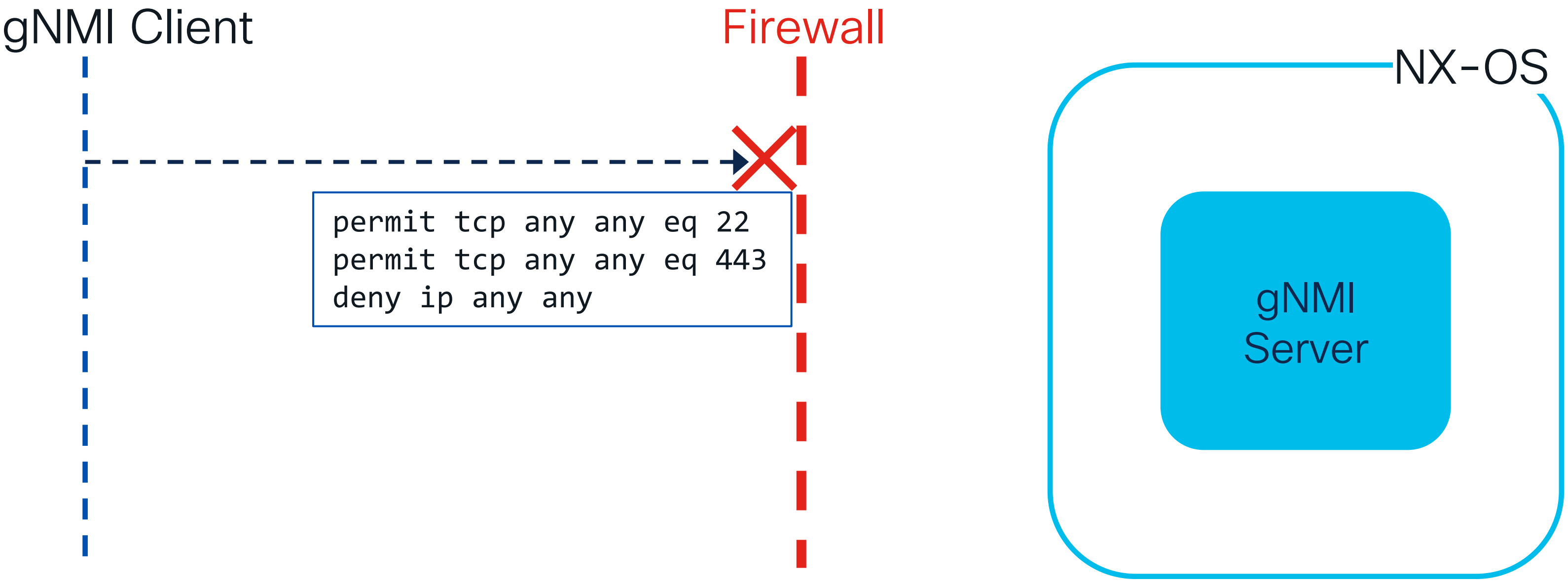
gRPC Network Management Interface

- Built on the gRPC framework
  - gRPC is based on HTTP/2
  - Specification of RPCs and behaviors for managing state on the network device
- Supports both configuration management and steaming telemetry
- Design to carry any tree-structured data
- Offers an alternative to NETCONF/RESTCONF

gNMI RPCs

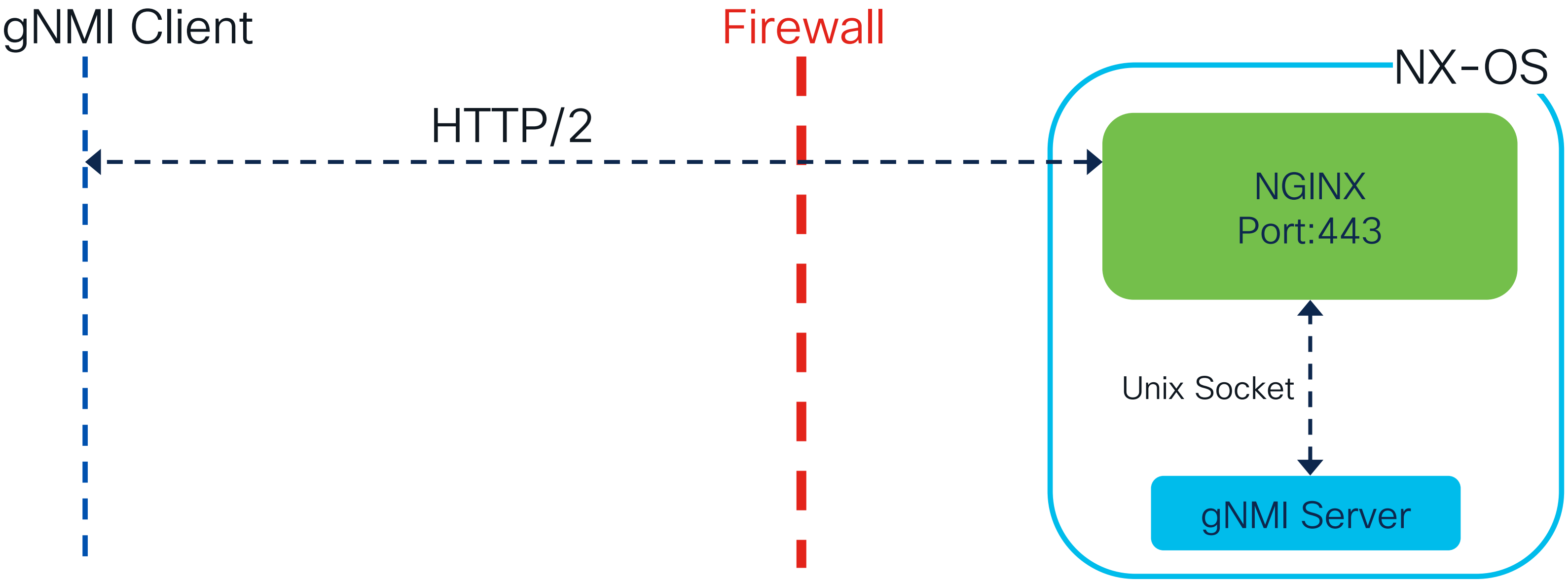
- Capabilities: Retrieve the capabilities supported by the target, usually happens during initial communication
- Get: Retrieve a snapshot of data from the target
- Set: Modify the state of data on the target
- Subscribe: Subscribe to a stream of values within the data tree

Most Firewalls Don’t Allow gNMI



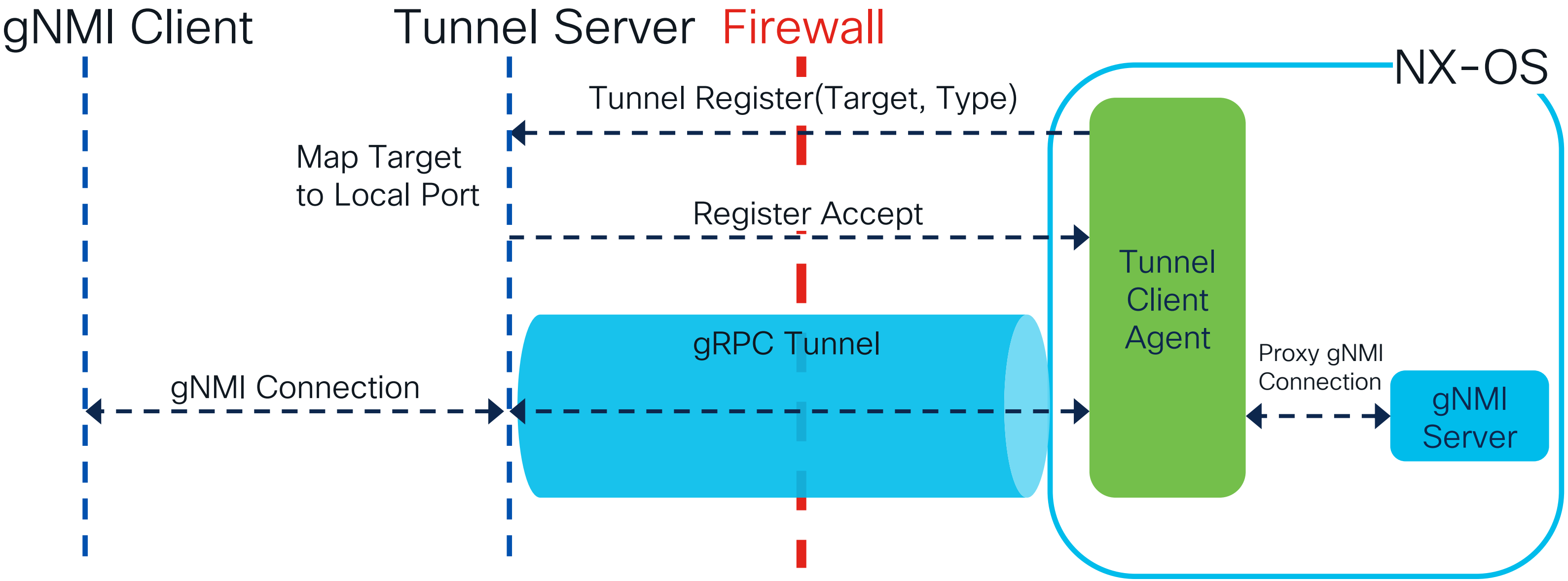
```
permit tcp any any eq 22
permit tcp any any eq 443
deny ip any any
```

NGINX Proxy



Shipping in 10.3(3)F

gRPC Tunnel



Shipping in
10.3(2)F

gNMI Implementation in NX-OS

Standard	gNMI in NX-OS 10.4.x is based on version 0.8.0
RPC Capabilities	All gNMI operations are supported since 9.3(5) Supports both ON_CHANGE and SAMPLE streaming mode `target_defined` is supported in 10.2(1)F `suppress_redundant` and `heartbeat_interval` is supported in 10.2(3)F
Security	TLS is mandatory, supports Mutual TLS
Data Model Encoding	Native and OpenConfig YANG model Supports GPB-KV and JSON as encoding Wild card is supported in 10.2(2)F

Open-Source Telemetry System

Open-Source Software Stack

Telemetry Collection Requires Three Pieces

1 Collection Agent

A service that understands the data collected from the device

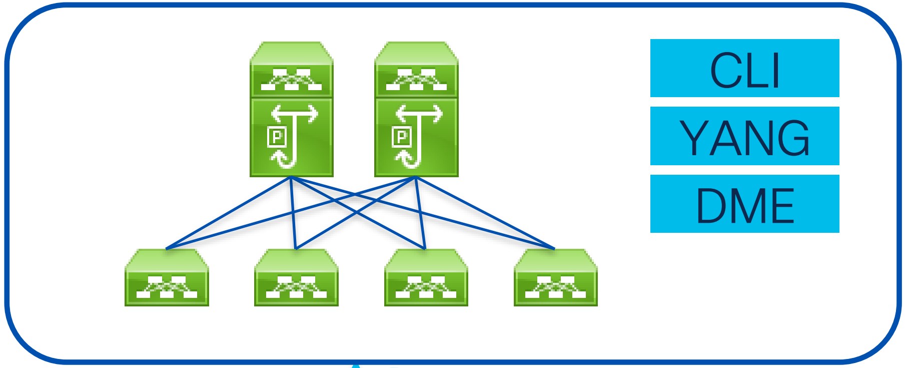
2 Time Series Database

A database with very precise time stamping that stores the collected data

3 Using Stored Data

Integrating the data with an automation system, or graphically displaying the data

Open-Source Software Stack



CLI

YANG

DME

gNMI
Dial-In

Telemetry
Dial-Out

Write Data

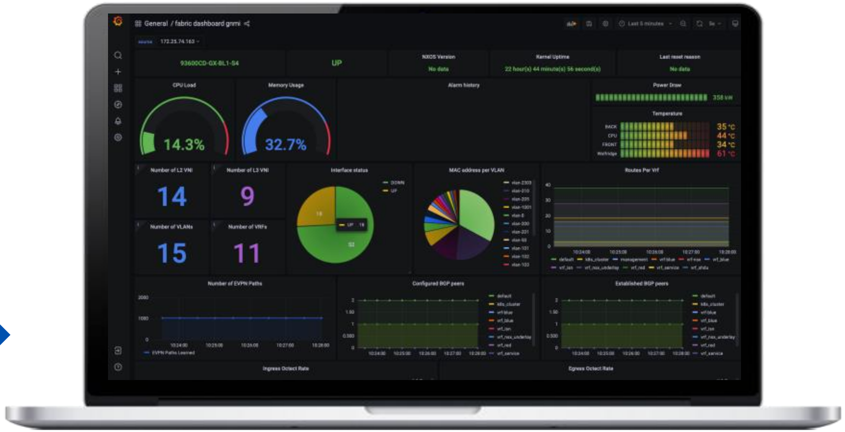
TSDB

telegraf

influxdb

Prometheus

Grafana



General / fabric dashboard gnmi

172.25.74.163

93600CD-GX-BL1-S4

UP

NXOS Version
No data

Kernel Uptime
22 hour(s) 44 minute(s) 56 second(s)

Last reset reason
No data

CPU Load
14.3%

Memory Usage
32.7%

Alarm history

Power Draw
358 kW

Temperature
BACK 35 °C
CPU 44 °C
FRONT 34 °C
Wetbridge 61 °C

Number of L2 VNI
14

Number of L3 VNI
9

Number of VLANs
15

Number of VRFs
11

https://github.com/dsx1123/telemetry_collector

Demo

Takeaways

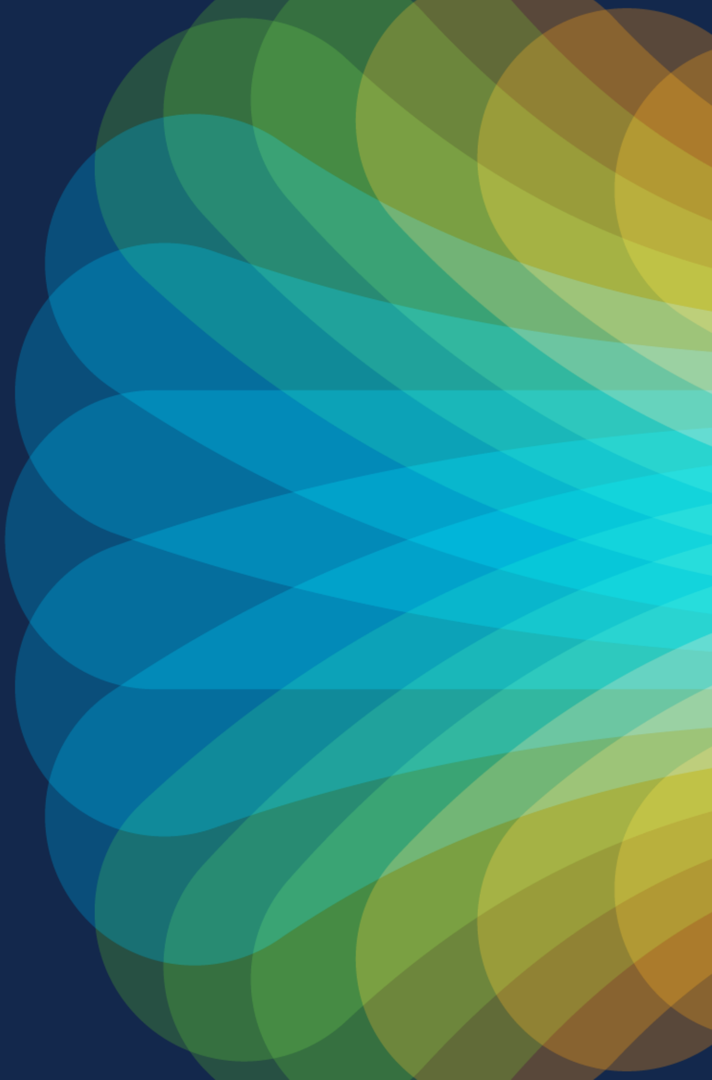
- NX-OS has a several choices for the data model and streaming transport options, customers can choose based on business requirements
- Most customers are interested in gNMI dial-in but there are pros and cons between dial-out and dial-in
- To optimize resource utilization, only stream what you need
- For efficiency, use GPB-KV when possible
- Use OpenConfig YANG models first, fall back to Native YANG model and DME when data is not available in OC YANG

Continue Your Journey

- DEVWKS-2135: Industry Standard Streaming Telemetry with Cisco NX-OS
- BRKDCN-2604: Model-Driven Programmability with Cisco NX-OS
- DEVNET-1677: Manage Your Cisco NX-OS Fabric with OpenConfig

The bridge to possible

Thank you



cisco Live!

Let’s go