Network Observability

Innovations at the IETF

23.05.2026, Thomas Graf – thomas.graf@swisscom.com

Picture: Apollo 8, December 24th 1968

swisscom

We have a dream

Digital Twin at your fingertips

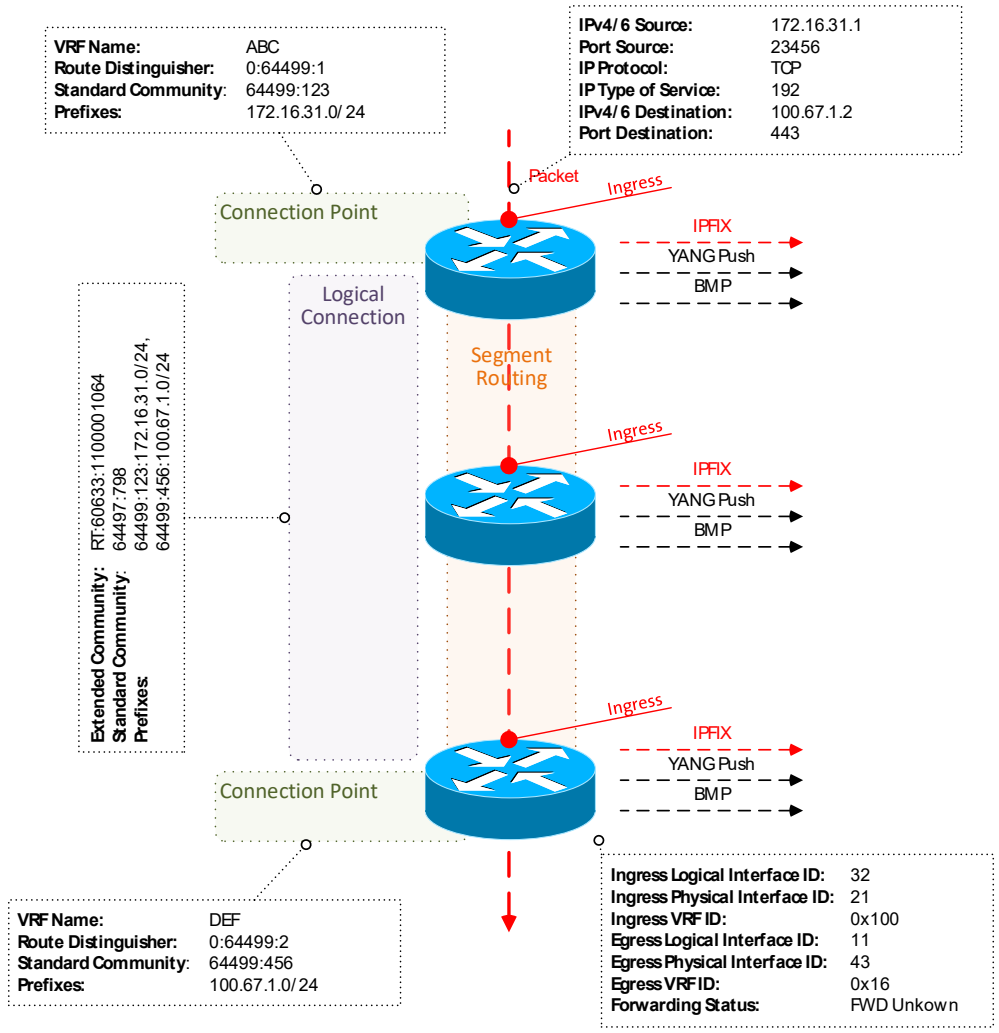
« Imagine that your entire life as network engineer you have logged into routers to perform show commands to get a glimpse into the current state of your networks. »

« Suddenly you see your colleague on the right querying the **current network state in seconds directly from a real-time data stream.** No access to routers needed. No databases needed. »

"NASA Mission Control Digital Twin in the 60s, Digital Twin is not Rocket Science"

Monitoring L3 VPN's with IPFIX, BMP and YANG-Push

From Connectivity Service to Realtime Network Observability



- **Connectivity Service perspective,** Connection Points are connected through Logical Connections.
- **From a BGP control-plane perspective,** IPv4/6 unicast prefixes in VRF's are tagged with BGP standard communities.
  - One BGP standard community to identify the Logical Connection. One BGP standard community to identify each Connection Point.
  - When IPv4/6 prefixes are exported from VRF's, a BGP route-distinguisher, BGP extended community route-targets and a SRv6 VPN SID for the IPv6 next-hop are allocated.
- **From a forwarding plane perspective,** when IPv4/6 unicast traffic is received from the edge at the SRv6 PE, a lookup is performed, the SRv6 VPN SID is obtained and IPv6 next-hop is added when forwarded to the core.
- **Swisscom collects** MPLS and SRv6 provider data plane, IPv4/6 unicast customer data-plane in IPFIX and at provider edge BGP VPNv4/6 unicast **in production** to perform real-time data correlation.

BGP Monitoring Protocol (BMP)

BGP RIB's with path decision and route-policy metadata

Integrates natively into the BGP process by mirroring BGP PDU's. Exports peering states and statistics and with route-monitoring the BGP RIB tables, BGP path decision and route-policy metadata.

BGP Monitoring Protocol (BMP)
https://datatracker.ietf.org/doc/html/rfc7854

Support for Adj-RIB-Out in BGP Monitoring Protocol
https://tools.ietf.org/html/rfc8671

Support for Local RIB in BGP Monitoring Protocol
https://datatracker.ietf.org/doc/html/rfc9069

New

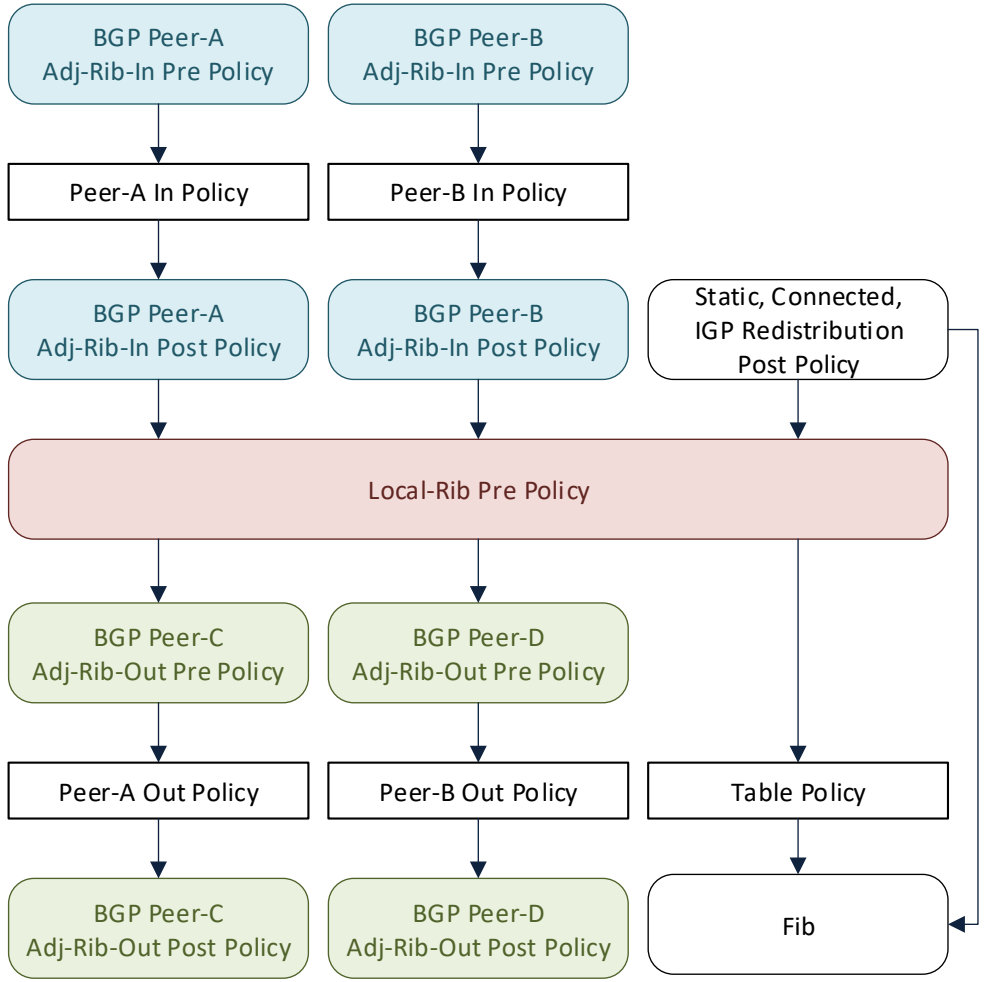
Advanced BMP Statistics Types
https://datatracker.ietf.org/doc/html/rfc9972

TLV support for BMP Route Monitoring and Peer Down Messages
https://tools.ietf.org/html/draft-ietf-grow-bmp-tlv

BMP Extension for Path Marking TLV
https://datatracker.ietf.org/doc/html/draft-ietf-grow-bmp-path-marking-tlv

Logging of routing events in BGP Monitoring Protocol
https://datatracker.ietf.org/doc/html/draft-ietf-grow-bmp-rel

BMP YANG Module
https://datatracker.ietf.org/doc/html/draft-ietf-grow-bmp-yang



BMP – Address Family Agnostic

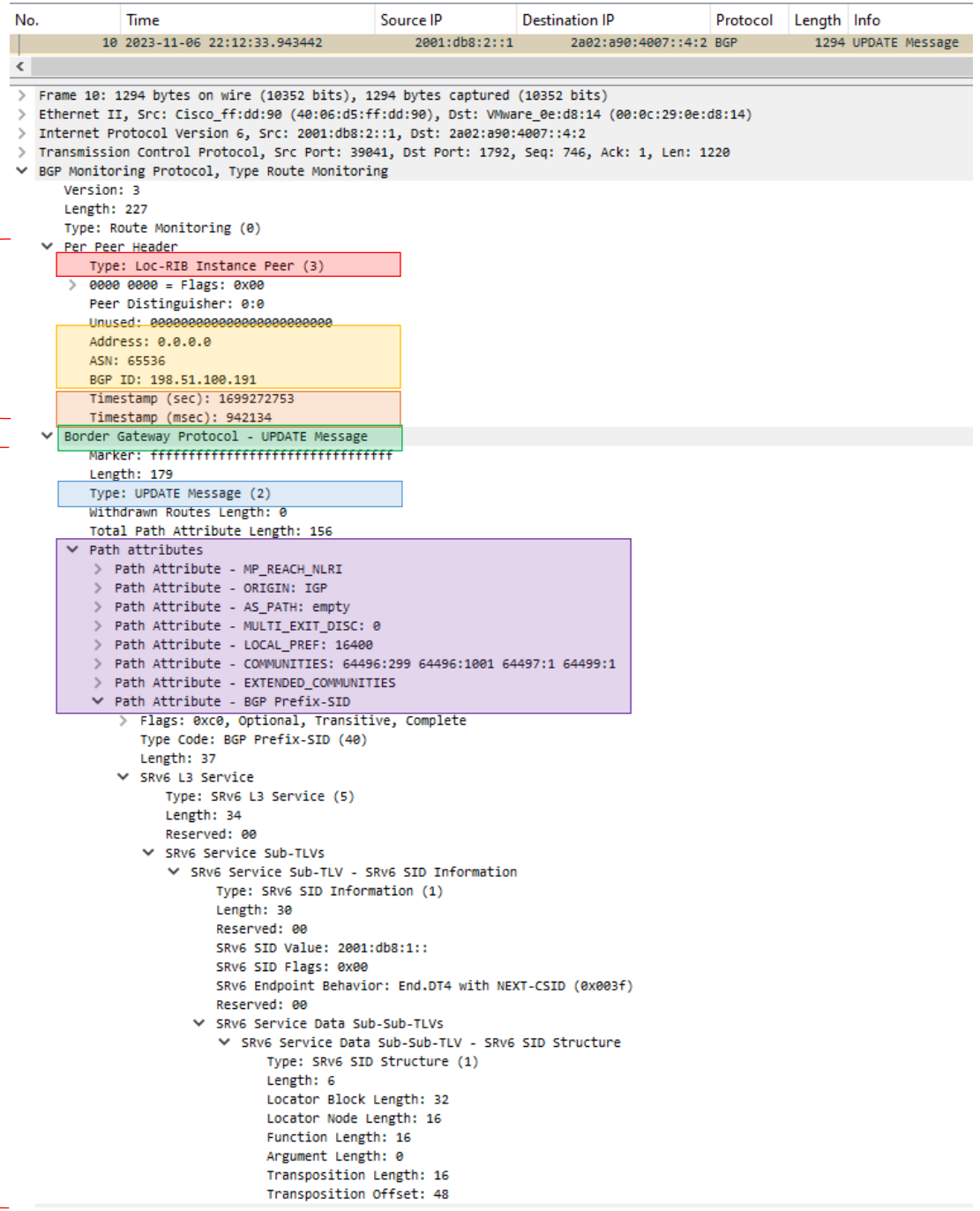
RFC 9069 provides Local RIB Visibility

> BMP Per Peer Header

Shows at which RIB (Adj-RIB In, Local or Adj-RIB Out, Pre or Post Policy) and from which Peering the BGP PDU at which time was obtained.

> Encapsulated BGP PDU

Shows the encapsulated BGP PDU. In case of BMP route-monitoring, it describes whether it was a topology update or withdrawal and for BGP community, NLRI and BGP Prefix SID path attributes.



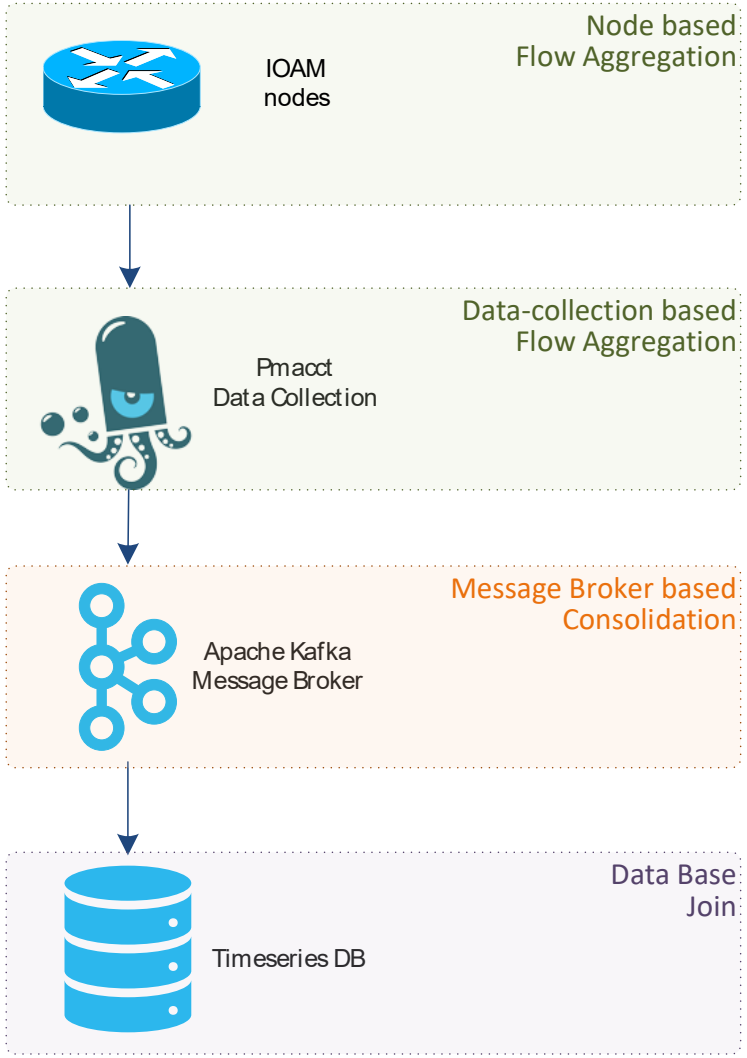
IPFIX Covering Segment Routing

For MPLS-SR, SRv6 and On-path Delay

SRv6 is commonly standardized, network vendors implementations are available, and network operators are at various stages in their deployments, missing data-plane visibility though.

Segment Routing coverage in IPFIX brings visibility for:

- Which routing protocol provided the label or IPv6 Segment in the SR domain.
- The active Segment where the packet is forwarded to in the SRv6 Domain.
- The Segment List where the packet is going to be forwarded throughout the SRv6 Domain.
- The Endpoint Behavior describing how the packet is being forwarded in the SRv6 Domain.
- The Min, Max and Average On-path delay at each hop in the SR domain.



Export of MPLS Segment Routing Label Type Information in IPFIX
https://datatracker.ietf.org/doc/html/rfc9160

Export of Segment Routing IPv6 Information in IPFIX
https://datatracker.ietf.org/doc/html/rfc9487

Export of Forwarding Path Delay in IPFIX
https://datatracker.ietf.org/doc/html/rfc9951

Segment Routing IPv6 Encapsulation

RFC 9487 provides IPFIX Visibility

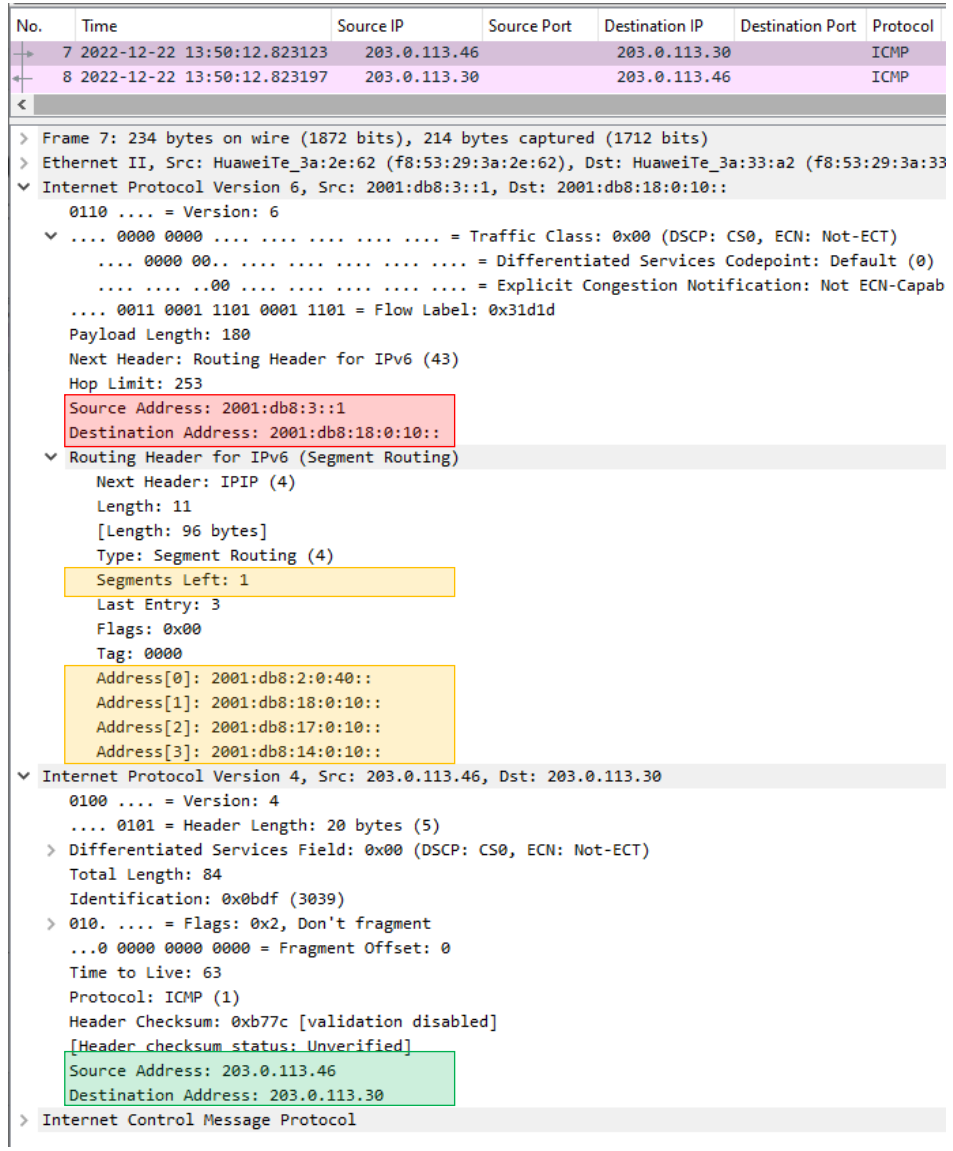
> Provider data-plane

Divided into an IPv6 and Segment Routing Header.

The IPv6 header shows from which PE to which next-hop it is being forwarded. The Segment Routing Header the list of segments this packet needs to pass through and points to the active segment.

> Customer data-plane

This is what we receive from the customer and encapsulate for transport through the SRv6 core.

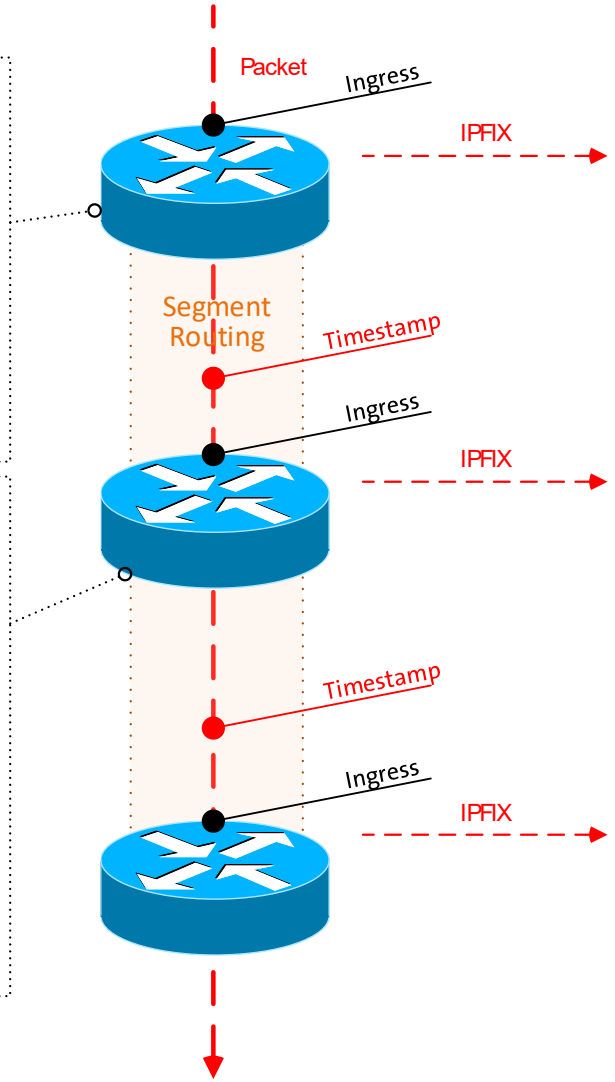


Measure On-Path Delay with Network Context

With draft-ietf-opsawg-ipfix-on-path-telemetry

IPv4/6 Source:	172.16.31.1
Port Source:	23456
IP Protocol:	TCP
IP Type of Service:	192
IPv4/6 Destination:	100.67.1.2
Port Destination:	443
Ingress Logical Interface ID:	32
Ingress Physical Interface ID:	21
Ingress VRF ID:	0x100
Egress Logical Interface ID:	11
Egress Physical Interface ID:	43
Egress VRF ID:	0x16
Forwarding Status:	FWD Unkown

IPv4/6 Source:	172.16.31.1
Port Source:	23456
IP Protocol:	TCP
IP Type of Service:	192
IPv4/6 Destination:	100.67.1.2
Port Destination:	443
Ingress Logical Interface ID:	32
Ingress Physical Interface ID:	21
Ingress VRF ID:	0x16
Egress Logical Interface ID:	11
Egress Physical Interface ID:	43
Egress VRF ID:	0x16
Forwarding Status:	FWD Unkown
SID List:	17001, 34002
Delay Min	1
Delay Sum	5
Delay Max	7



- Packets are **captured** ingress with an **optional sampler**, data plane dimensions **extracted**, **enriched** with management and control plane dimensions and added with a unique **flow ID** to a flow cache on the node for aggregation.
- **A direct export marking bit and optionally a timestamp is added** to the packet when entering the OAM domain by leveraging IOAM Direct Export (RFC 9378) or E2E Option Type (RFC 9197) or alternatively Enhanced Alternate Marking (RFC 9341, draft-zhou-ippm-enhanced-alternate-marking).
- Each subsequent packet for the same flow increases byte and packet count. Each new flow creates a new flow ID in the flow cache.
- **At each node** in transit or only at the decapsulation node, **delay is calculated** by comparing the observation timestamp in the packet and when packet is received. **Delay is populated into the flow cache together with packet and byte count** as defined in RFC 9951.

From YANG-Push to Network Analytics

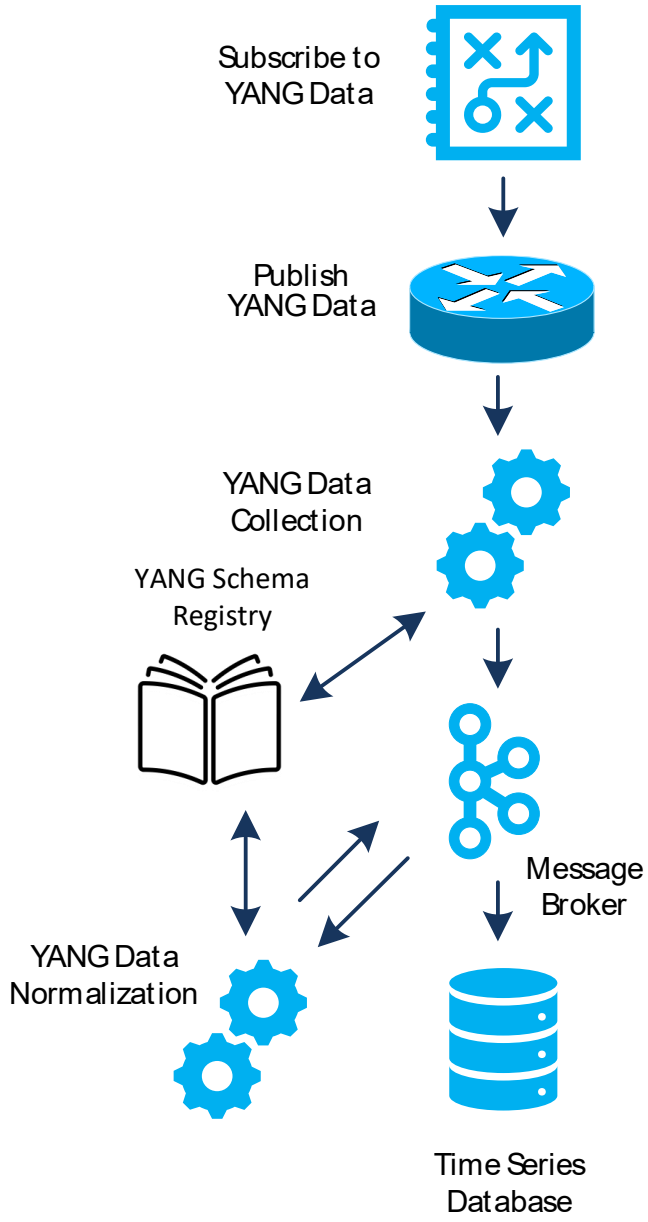
Aiming for an automated data processing pipeline

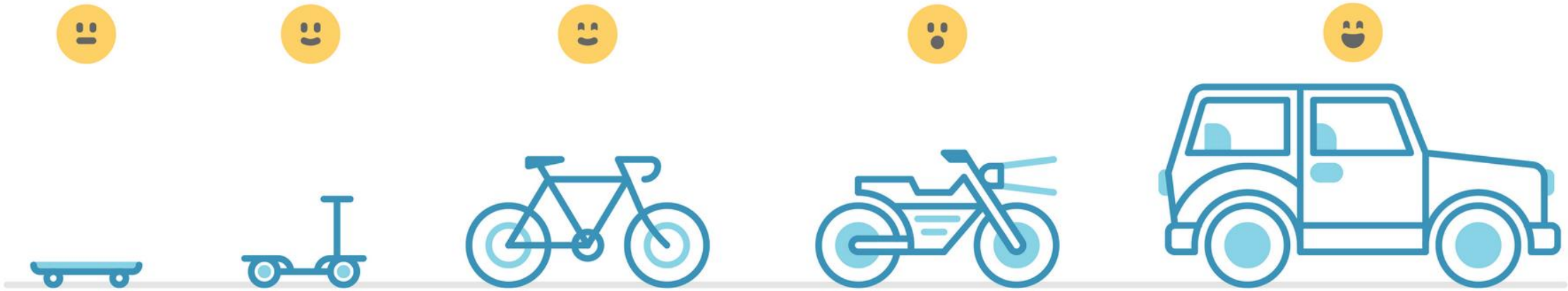
A network operator aims for:

- An **automated data processing pipeline** which starts with YANG-Push, consolidates at Data Mesh and ends at Network Analytics.
- Operational metrics where **standards organizations define semantics.**
- Analytical metrics where **network operators gain actionable insights.**

We achieve this by integrating YANG-Push into Data Mesh to:

- Produce metrics from networks **with timestamps when network events were observed.**
- Hostname, publisher ID and sequence numbers help us to understand **from where metrics were exported and measure its delay and loss.**
- Forward **metrics unchanged** from networks
- **Learn semantics** from networks and validate messages.
- **Control semantic** changes end to end.





Today, subscribing to a YANG datastore, publishing a YANG modeled notifications message from the network and viewing the data in a time series database, **manual labor is needed to perform data transformation** to make a message broker and its data processing components with YANG notifications interoperable.

State of the Union

From data mess to data mesh

IETF YANG-Push

A 23 years journey

2002 — IAB Workshop

Defines operators' requirements in RFC 3535 to lifecycle CLI and SNMP. YANG, Netconf and Restconf development started.

2010 — YANG 1.0

Specified in RFC 6020. 1.1 in RFC 7950.

2015 — IETF YANG-Push Specification Started

Development of RFC 8639 and RFC 8641 started at IETF NETCONF.

2017 — gNMI

gNMI was presented to IETF NETCONF and implementations started at major network vendors.

2019 — IETF YANG-Push Specification Finished

Development of RFC 8639 and RFC 8641 concluded at IETF NETCONF without any major network vendor implementations.

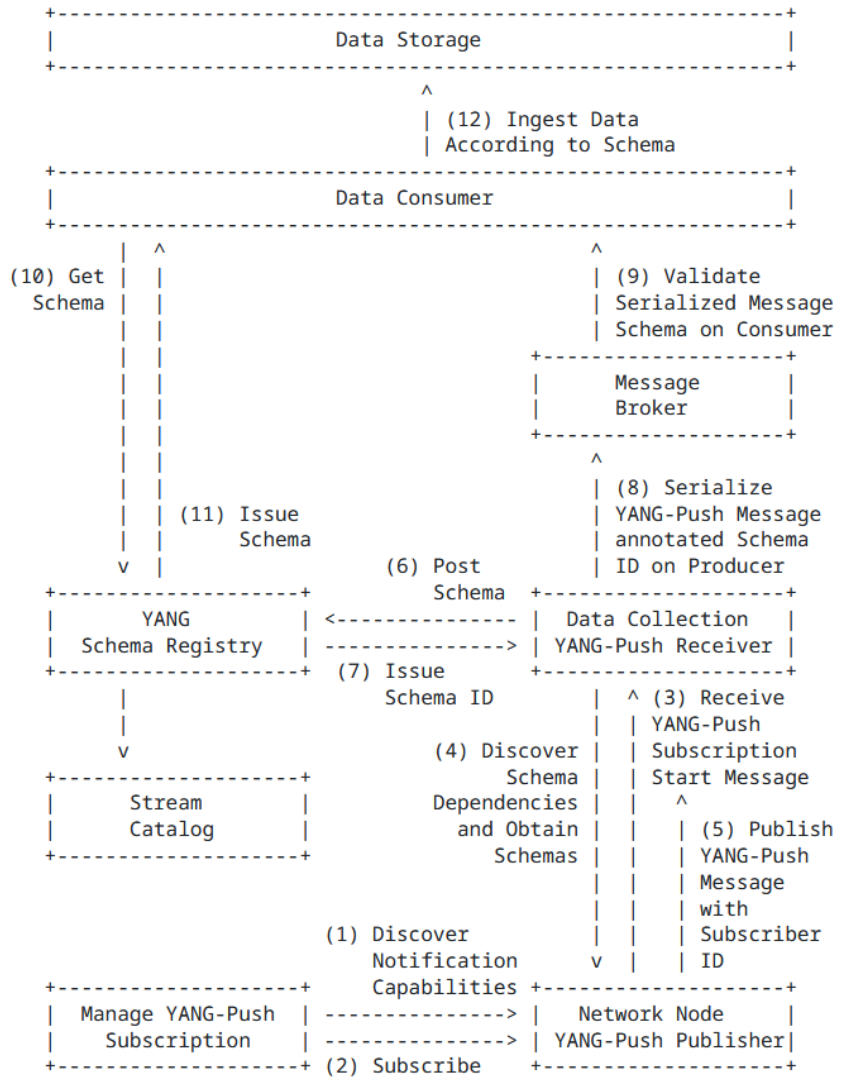
2022 — Data Mesh Integration

Vendor-specific implementations and IETF YANG-Push are hard to manage. New requirements emerged for integrating with the message broker and an automated data processing chain. New specifications are proposed to resolve these challenges.

2024 — IETF YANG-Push Major Implementations Started

Questions arise. Proposing a simplified IETF YANG-Push and an Agile Incremental Driven Development.

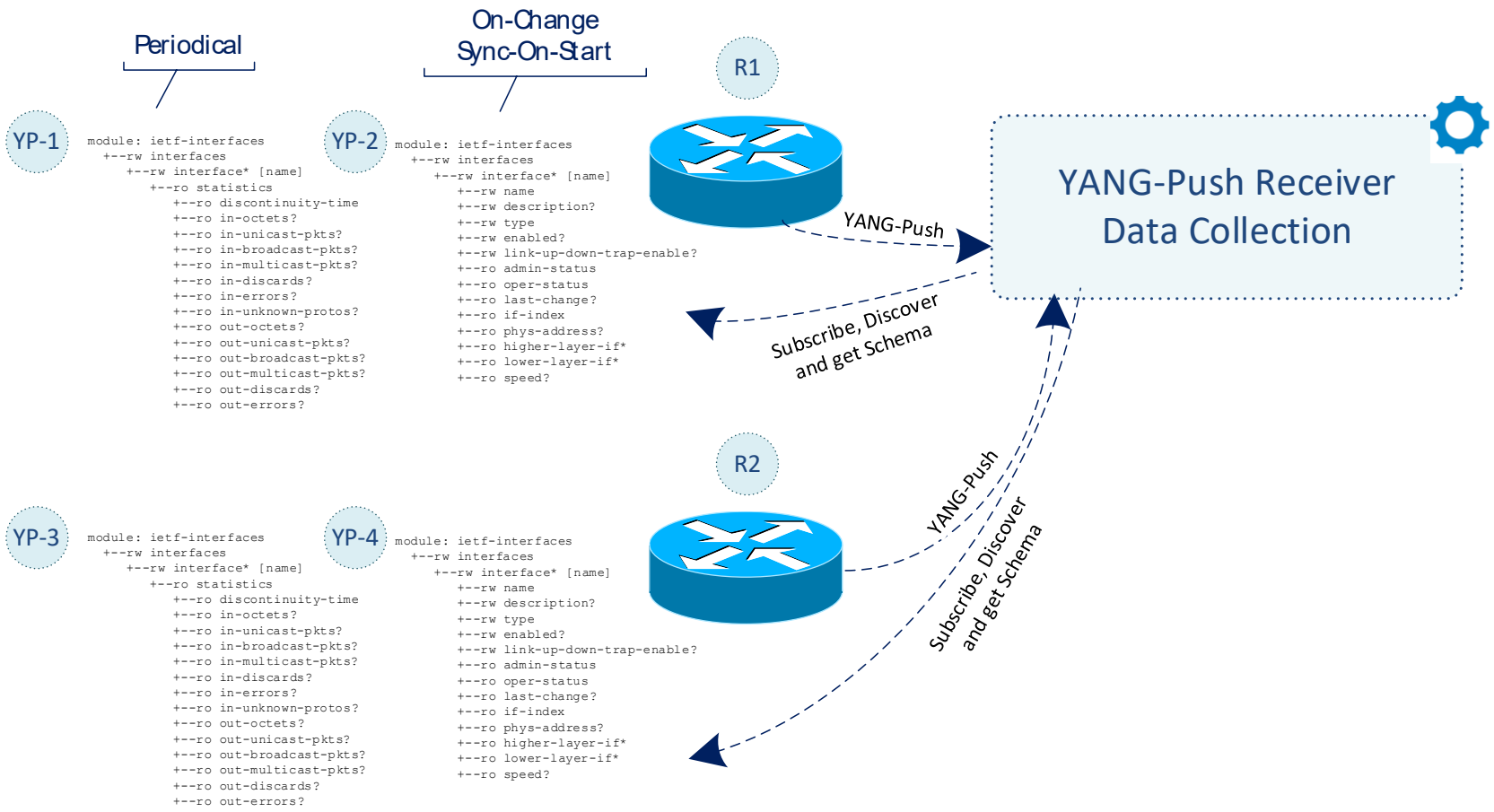
Elements of the Architecture
Workflow Diagram



- **Network Orchestration** subscribes to YANG datastore.
- **Network Node** informs Data Collection on subscription state and publishes YANG metrics with YANG-Push.
- **Data Collection** obtains for each subscription the YANG module dependencies and the YANG modules on the network node, registers it in the YANG Schema Registry and prefixes the forwarded YANG notifications with the obtained schema ID.
- **YANG Schema Registry** issues for a Message Broker subject a schema ID for each new schema tree, compares a new schema tree with an existing and versions it.
- **Data Consumer** consumes YANG-Push notifications from Message Broker, obtains schema tree from YANG schema registry, validates YANG notifications against schema and uses schema to populate into database table.
- **Architecture Details:** [draft-ietf-nmop-yang-message-broker-integration]

YANG-Push

Discover and Subscribe to YANG metrics



From discovering YANG-Push subscription capabilities defined in RFC 9196, subscribing interesting metrics periodical (**statistics**), on-change (**state changes**) or on-change with sync-on-start (**states**) defined in RFC 8641.

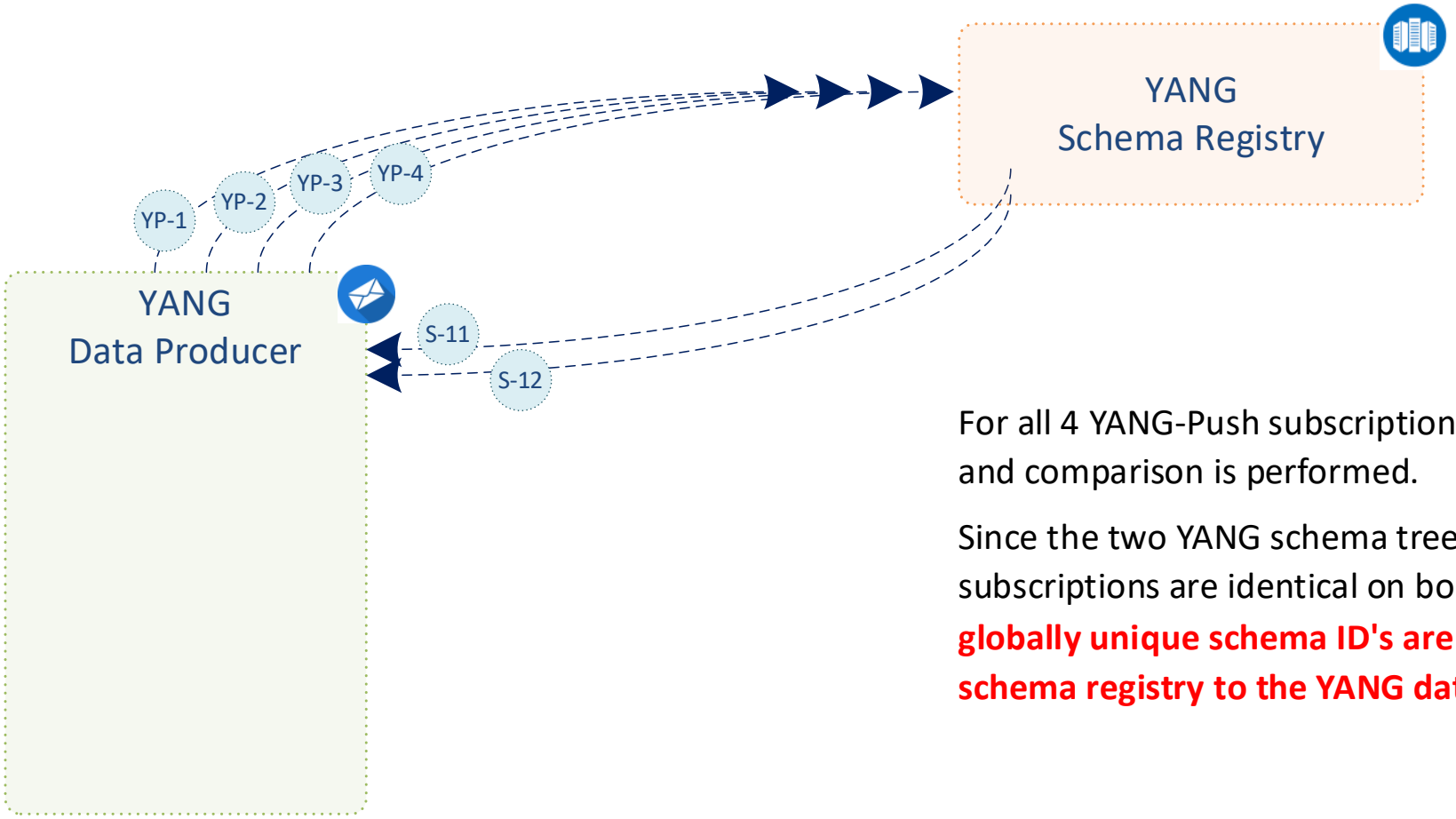
Each subscription refers to network node, datastore (RFC 8342) and a schema tree.

In this RFC 8343 example ietf-interface statistics are subscribed periodically and ietf-interface states on-change sync-on-start. **YANG-Push subscription ID's are per network node significant.**

Data Collection obtains for each subscription the YANG schema tree by leveraging <get-schema> (RFC 6022), YANG Library (RFC 8525) and draft-ietf-netconf-yang-library-augmentedby.

YANG Schema Registry

From 4 subscription ID's to 2 schema ID's

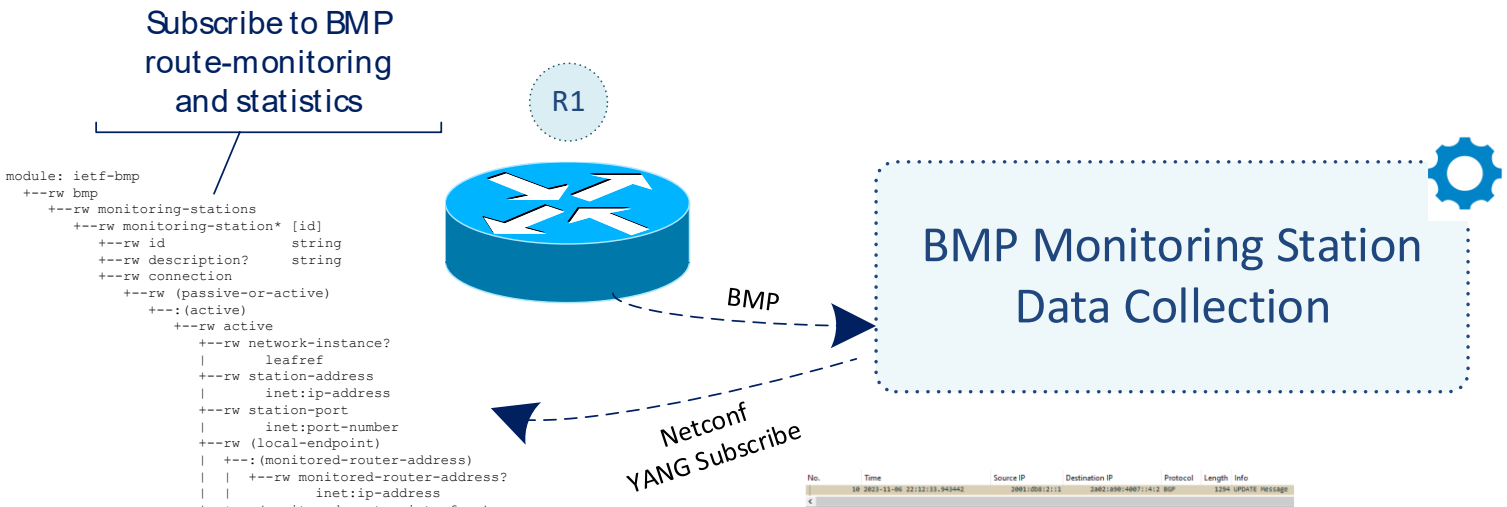


For all 4 YANG-Push subscriptions, YANG schema registration and comparison is performed.

Since the two YANG schema trees for both YANG-Push subscriptions are identical on both network nodes, **two YANG globally unique schema ID's are being issued from the YANG schema registry to the YANG data producer.**

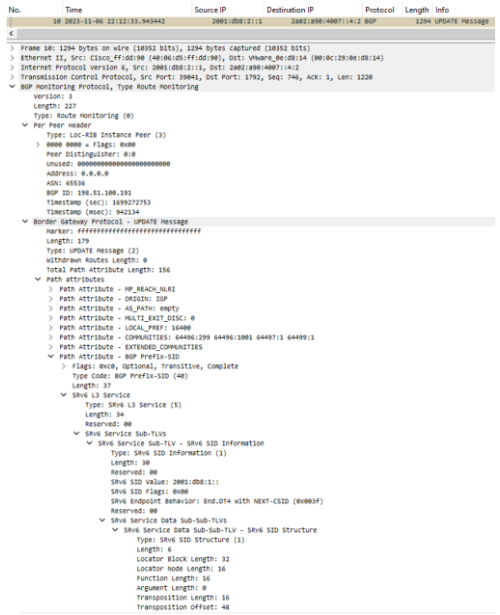
BGP Monitoring Protocol

Subscribe to BGP metrics




```
module: ietf-bmp
  +--rw bmp
     +--rw monitoring-stations
        +--rw monitoring-station* [id]
           +--rw id                string
           +--rw description?      string
           +--rw connection
              +--rw (passive-or-active)
                 +--:(active)
                    +--rw active
                       +--rw network-instance?
                       |       leafref
                       +--rw station-address
                       |       inet:ip-address
                       +--rw station-port
                       |       inet:port-number
                       +--rw (local-endpoint)
                       |  +--:(monitored-router-address)
                       |  |  +--rw monitored-router-address?
                       |  |          inet:ip-address
                       |  +--:(monitored-router-interface)
                       |     +--rw monitored-router-interface?
                       |             if:interface-ref
                       +--rw monitored-router-port?
                               inet:port-number
```

```
module: ietf-bmp
  +--rw bmp
     +--rw monitoring-stations
        +--rw monitoring-station* [id]
           +--rw id                string
           +--rw description?      string
           +--rw bmp-data
              +--rw initiation-message?   string
              +--rw statistics-report!
              |  +--rw statistics-interval    uint32
              +--rw route-monitoring
                 +--rw network-instance-configuration
                    +--rw network-instances
                       +--rw network-instance* [id]
                          +--rw id                    leafref
                          +--rw enabled?              boolean
                          +--rw local-rib
                             +--rw address-families
                                +--rw address-family* [id]
                                   +--rw id          identityref
                                   +--rw filters
                                      +--rw policy-filter
                                             {bmp-filter-based-on-route-policy}?
                                        +--rw export-policy*
                                        |       leafref
                                        +--rw default-export-policy?
                                                rt-pol:default-policy-type
```



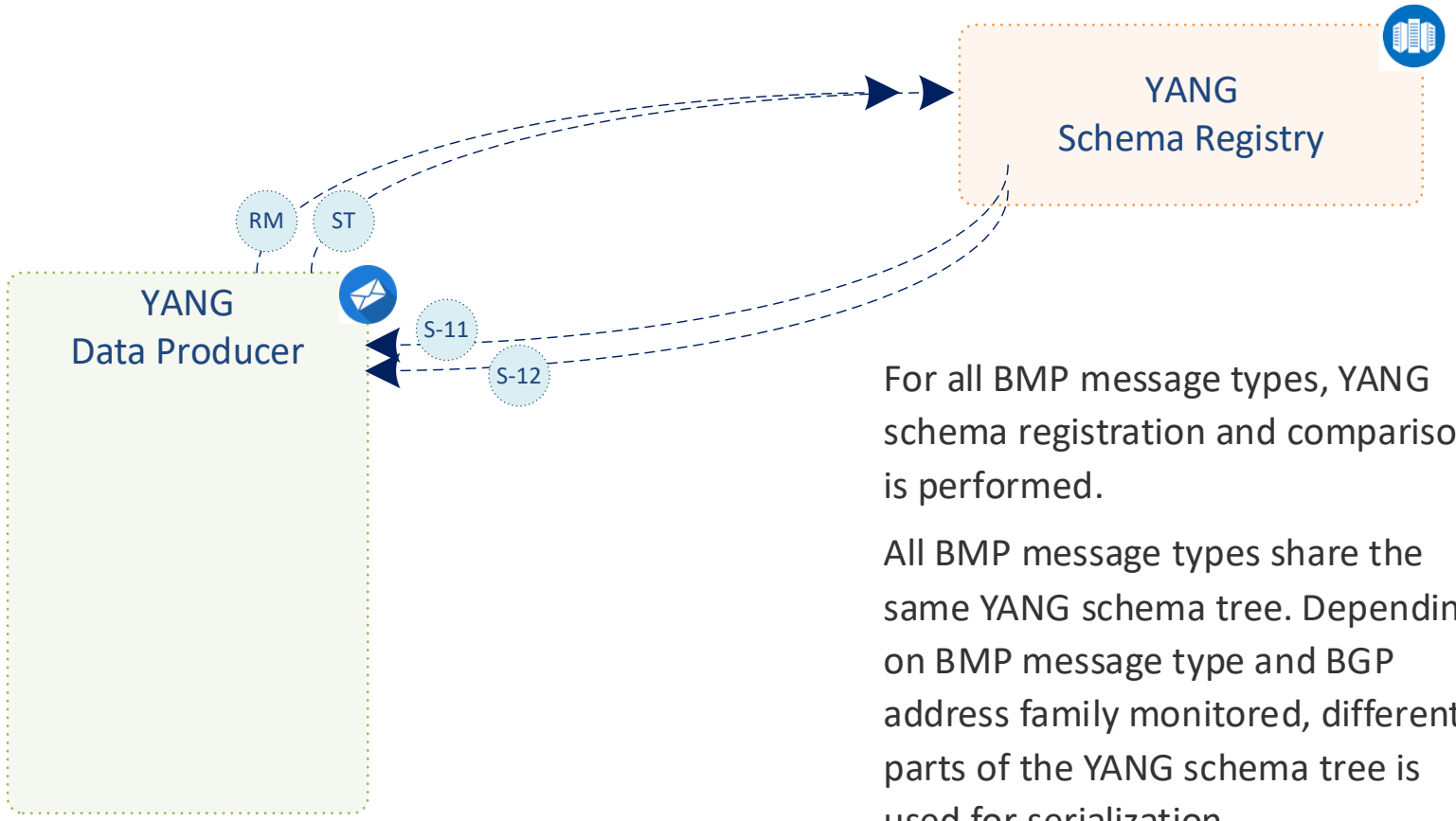
Through Netconf/Restconf and draft-ietf-grow-bmp-yang defined YANG mopdules BMP metrics are subscribed.

With BMPv3 defined in RFC 7854 resp. RFC 8671 and RFC 9069 information, peer-up, peer-down, route-monitoring, route-mirroring and statistic messages are exported to BMP monitoring station.

At the BMP monitoring station, the received BMP messages are **transformed to YANG** according to draft-netana-nmop-message-broker-bmp-telemetry-msg defined YANG modules.

YANG Schema Registry

Register BMP Telemetry Message Schemas



For all BMP message types, YANG schema registration and comparison is performed.

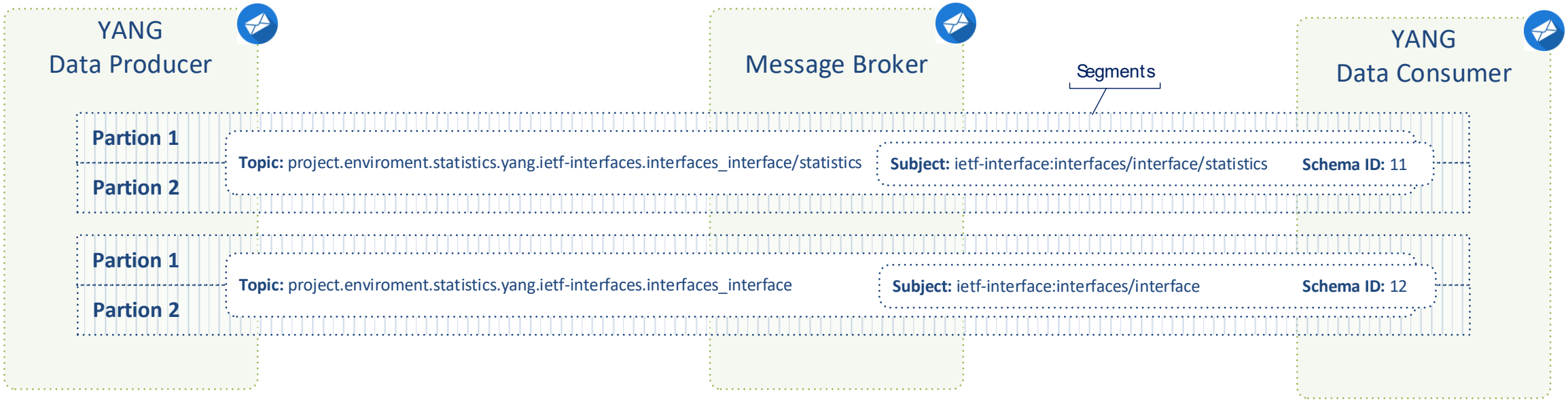
All BMP message types share the same YANG schema tree. Depending on BMP message type and BGP address family monitored, different parts of the YANG schema tree is used for serialization.

```
module: ietf-bmp-telemetry-message

  structure message:
    +-- version?          uint8
    +-- (message-type)?
       +--:(route-monitoring)
          +-- route-monitoring
             +-- peer-type?            peer-type
             +-- peer-flags?           uint8
             +-- peer-distinguisher?   rt-types:route-distinguisher
             +-- peer-address?         inet:ip-address
             +-- peer-as?              uint32
             +-- timestamp?            yang:date-and-time
             +-- afi-safi-type         identityref
             +-- rib-entry
                +-- (afi-safi)
                   +--:(ipv4-unicast)
                      |  +-- ipv4-unicast
                      |     +-- (rib-type)
                      |        +--:(loc-rib)
                      |        |  +-- loc-rib
                      |        |     +-- route
                      |        |        +-- prefix                    inet:ipv4-prefix
                      |        |        +-- origin?                   union
                      |        |        +-- path-id?                  uint32
                      |        |        +-- attributes
                      |        |        |  +-- origin?                bt:bgp-origin-attr-type
                      |        |        |  +-- as-path
                      |        |        |  |  +-- segment*
                      |        |        |  |     +-- type?     identityref
                      |        |        |  |     +-- member*   inet:as-number
                      |        |        |  +-- next-hop?              inet:ip-address
                      |        |        |  +-- link-local-next-hop?   inet:ipv6-address
                      |        |        |  +-- med?                   uint32
                      |        |        |  +-- local-pref?            uint32
                      |        |        |  +-- as4-path
                      |        |        |  |  +-- segment*
                      |        |        |  |     +-- type?     identityref
                      |        |        |  |     +-- member*   inet:as-number
                      |        |        |  +-- aggregator
                      |        |        |  |  +-- as?           inet:as-number
                      |        |        |  |  +-- identifier?   yang:dotted-quad
                      |        |        |  +-- aggregator4
                      |        |        |  |  +-- as4?          inet:as-number
                      |        |        |  |  +-- identifier?   yang:dotted-quad
                      |        |        |  +-- atomic-aggregate?      boolean
                      |        |        |  +-- originator-id?         yang:dotted-quad
                      |        |        |  +-- cluster-list*          yang:dotted-quad
                      |        |        |  +-- aigp-metric?           uint64
                      |        |        +-- community*                union
                      |        |        +-- ext-community*            bct:bgp-ext-community-type
                      |        |        +-- ext-community-raw*        string
                      |        |        +-- ipv6-ext-community*       bct:bgp-ipv6-ext-community-type
                      |        |        +-- ipv6-ext-community-raw*   string
                      |        |        +-- large-community*          bct:bgp-large-community-type
                      |        |        +-- last-modified?            yang:timeticks
                      |        |        +-- eligible-route?           boolean
                      |        |        +-- ineligible-reason?        identityref
                      |        |        +-- unknown-attributes
                      |        |        |  +-- unknown-attribute* [attr-type]
                      |        |        |     +-- attr-type     uint8
                      |        |        |     +-- optional?     boolean
                      |        |        |     +-- transitive?   boolean
                      |        |        |     +-- partial?      boolean
                      |        |        |     +-- extended?     boolean
                      |        |        |     +-- attr-len?     uint16
                      |        |        |     +-- attr-value?   binary
                      |        |        +-- reject-reason?            union
```

Message Broker

Topics, Subjects, Partitions, Segments and Message Keys



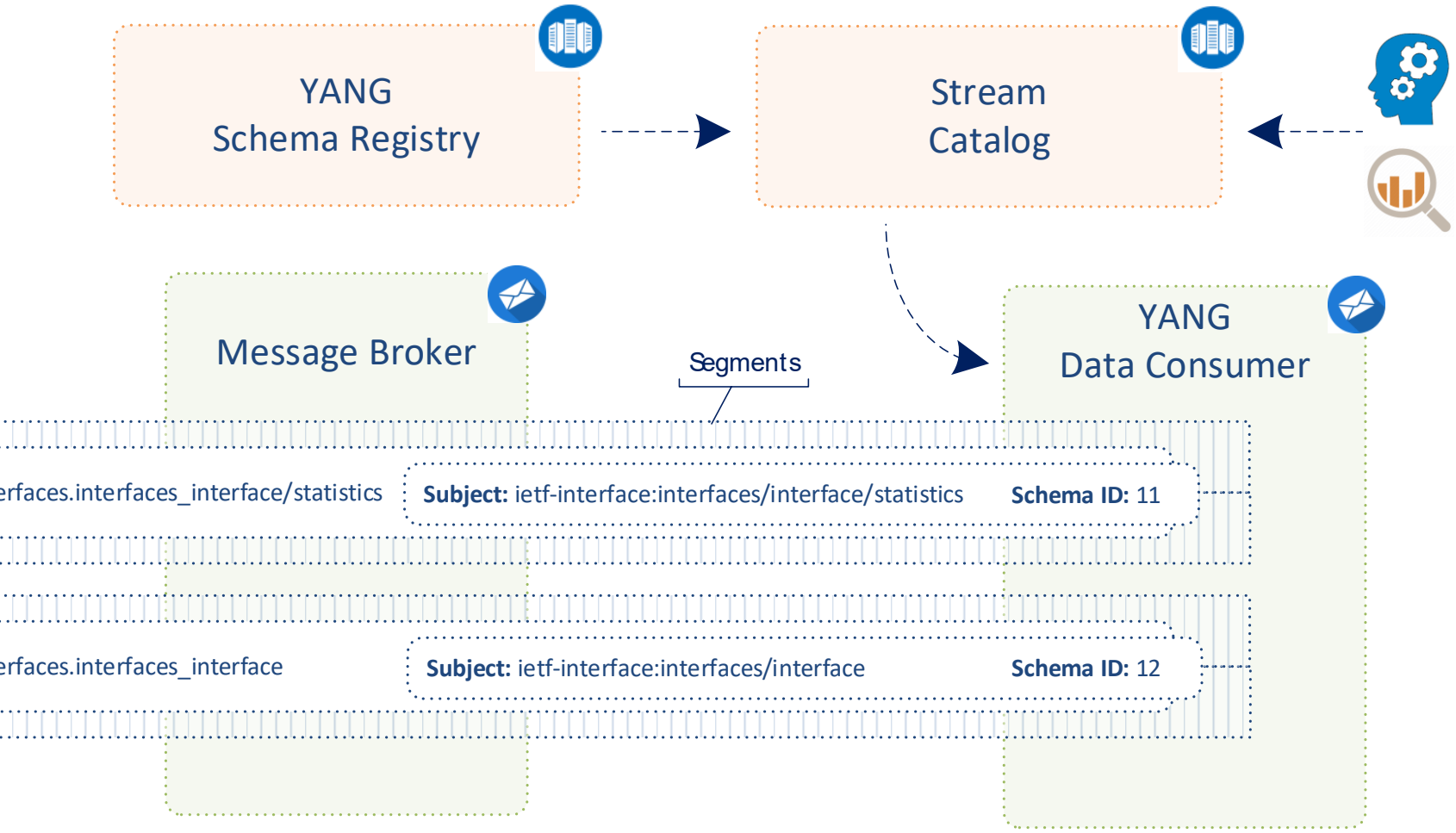
The YANG data producer creates for each YANG schema a new message broker topic, a message key and defines the number of partitions being used for the topic.

It serializes the message with the previously generated message key and message content according to draft-ietf-nmop-message-broker-telemetry-message and draft-netana-nmop-message-broker-bmp-telemetry-msg augment.

Each message is prefixed with the previously obtained schema ID representing a unique message subject. The messages are distributed according to the hashed message key across the partitions into continuous segments.

YANG Data Consumption

Discover and Subscribe to YANG metrics



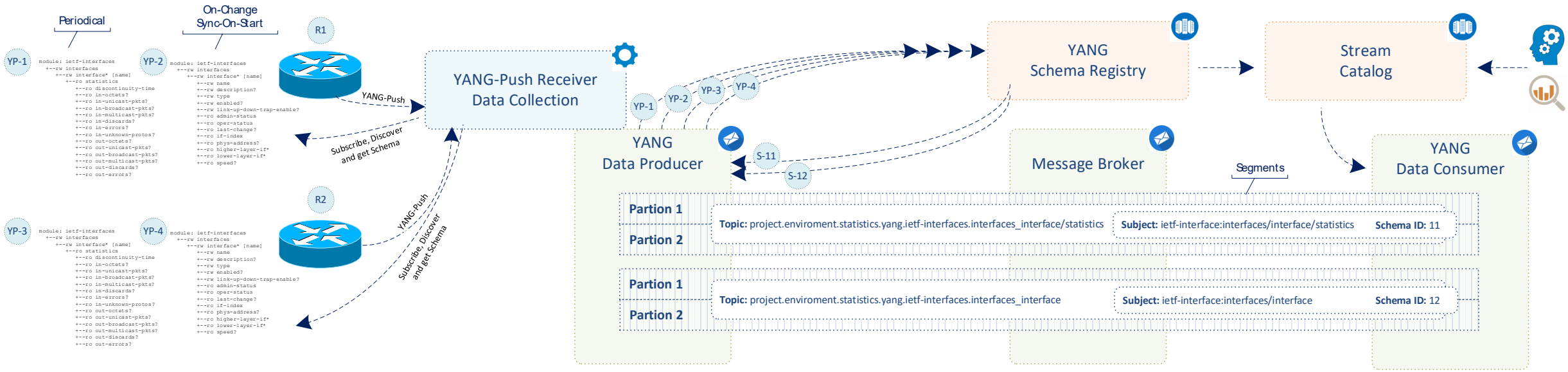
A user or AI application/agent subscribes discovers through the stream catalog interesting metrics and subscribes to message broker topic.

More than one topic can be consumed at once by using a wildcard such as: project.enviroment.states.yang.* to consume all YANG state metrics.

The consumer hashes the message key and applies modulo with the number of partitions to determine the partition it needs to consume from to obtain messages with desired message key.

YANG Converged Network Telemetry Data Processing Pipeline

From network nodes to message broker nodes



At the YANG data producer, thanks to the YANG awareness to the YANG data producer and the YANG schema registry, **for each unique YANG schema tree a message broker topic and subject is being created, and messages are being indexed and hashed according to the message key**.

At the message broker, thanks to the dedicated topic per YANG schema and its message key, **YANG state metrics can be compacted to the latest state; reducing the number of stored messages.**

At the YANG data consumer, thanks to the YANG awareness of the stream catalog and consumer, a specific topic, partition and subject can be consumed for the interested YANG metrics; **reducing the number of messages to be consumed.**


Yang
Kafka

Addressing YANG Specification and Integration Gaps

10 documents at IETF NMOP, NETCONF and NETMOD

YANG-Push Transport Gaps:

IESG • UDP-based Transport for Configured Subscriptions
draft-ietf-netconf-udp-notif

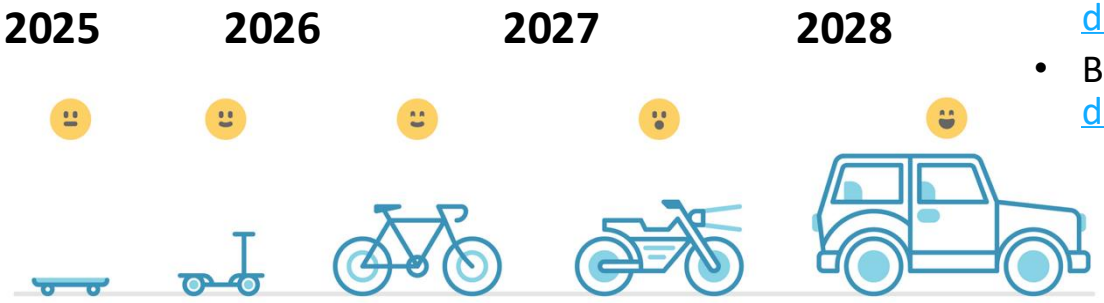
IESG • Subscription to Distributed Notifications
draft-ietf-netconf-distributed-notif

YANG-Push Specifications Gaps:

IESG • YANG Notification Transport Capabilities
draft-ietf-netconf-yp-transport-capabilities

IESG • Extensible YANG model for YANG-Push Notifications
draft-ietf-netconf-notif-envelope

• Validating anydata in YANG Library context
draft-ietf-netmod-yang-anydata-validation



YANG-Push Integration Gaps and Arch:

IESG • Support of Versioning in YANG Notifications Subscription
draft-ietf-netconf-yang-notifications-versioning

IESG • Augmented-by Addition into the IETF-YANG-Library
draft-ietf-netconf-yang-library-augmentation

YANG-Push Simplification:

• YANG-Push Operational Data Observability Enhancements
draft-wilton-netconf-yp-observability

YANG-Push Message Broker:

• An Architecture for YANG-Push to Message Broker Integration
draft-ietf-nmop-yang-message-broker-integration

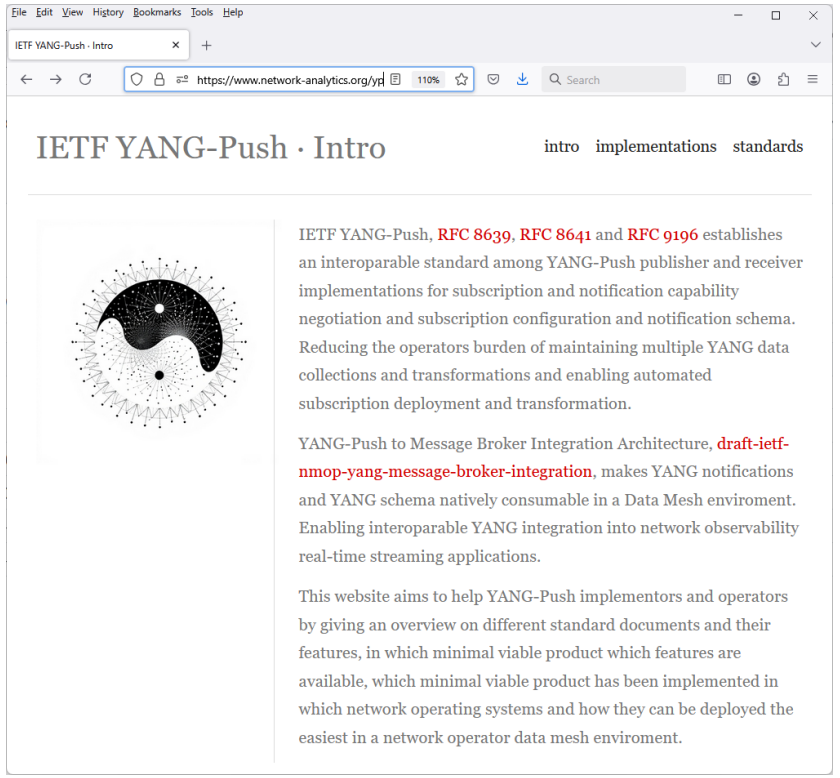
• Extensible YANG Model for Network Telemetry Notifications
draft-ietf-nmop-message-broker-telemetry-message

• YANG Message Keys for Message Broker Integration
draft-ietf-nmop-yang-message-broker-message-key

• BMP YANG Model for Network Telemetry Messages
draft-netana-nmop-message-broker-bmp-telemetry-msg

IETF 125 Hackathon

Test and Development Plan and Software



Test Plan

Validate and verify

- 5 YANG-Push Publishers
- 2 YANG-Push Receivers
- 2 YANG-Push Network Telemetry Message
- 1 YANG Message Broker Producer and Schema Registry
- 3 YANG Message Broker Consumers

implementation in YANG data integration automation. Subscribe to YANG data on YANG-Publisher, obtain and register all YANG modules necessary to build YANG schema tree, register YANG schemas to Schema Registry and verify YANG notifications against scheme trees and produce and consume from Message Broker.

https://www.network-analytics.org/yp/how-to-deploy.html

Software

- YANG-Push Publisher - Cisco IOS XR, 6WIND VSR, Huawei NE (Router) and MA (OLT) and Arrcus Arcos
- YANG-Push Receiver – Netgauze and Pmacct
- YANG Message Broker Producer – Netgauze
- YANG Message Broker Consumer – Netgauze, Ciena Blueplanet UAA Workflow Engine, Cisco Crosswork Ingestion Service
- YANG Schema Registry – Apache Kafka YANG Plugin
- Wireshark - udp-notif dissector

YANG-Push Publisher

IETF 125 Implementation Status

	6WIND VSR	Huawei NE	Huawei MA	Cisco IOS XR	Arrcus Arcos	Open-Source
RFC 8639 YANG-Push Subscription	✓	✓	✓	✓	✓	
RFC 8641 YANG-Push Notification	✓	✓	✓	✓	✓	
draft-ietf-netconf-udp-notif	✓	✓	✓	✓	✓	✓
draft-ietf-netconf-distributed-notif	✓	✓	✓			
draft-ietf-netconf-notif-envelope	✓	✓	✓	✓	✓	
draft-ietf-netconf-yang-notifications-versioning	✓	✓	✓	✓		
RFC 8525 YANG Library	✓	✓	✓	✓		
draft-ietf-netconf-yang-library-augmentation	✓	✓	✓	✓		✓
RFC 9196 System and Notification Capabilities			✓	P		
draft-netana-netconf-yp-transport-capabilities			✓	✓	✓	
RFC 9254 CBOR Named Identifiers	✓					

NetGauze
Pmacct
udp-notif-c-collector
yang-library-augmentedby

Green marked describes new capability at IETF 125. "P" to partially implemented.

RFC 8348 – ietf-hardware.yang

YANG Data Model for Hardware Management

```
module: ietf-hardware
  +--rw hardware
     +--ro last-change?   yang:date-and-time
     +--rw component* [name]
        +--rw name                string
        +--rw class               identityref
        +--ro physical-index?     int32 {entity-mib}?
        +--ro description?        string
        +--rw parent?             -> ../../component/name
        +--rw parent-rel-pos?     int32
        +--ro contains-child*     -> ../../component/name
        +--ro hardware-rev?       string
        +--ro firmware-rev?       string
        +--ro software-rev?       string
        +--ro serial-num?         string
        +--ro mfg-name?           string
        +--ro model-name?         string
        +--rw alias?              string
        +--rw asset-id?           string
        +--ro is-fru?             boolean
        +--ro mfg-date?           yang:date-and-time
        +--rw uri*                inet:uri
        +--ro uuid?               yang:uuid
        +--rw state {hardware-state}?
        |  +--ro state-last-changed?   yang:date-and-time
        |  +--rw admin-state?          admin-state
        |  +--ro oper-state?           oper-state
        |  +--ro usage-state?          usage-state
        |  +--ro alarm-state?          alarm-state
        |  +--ro standby-state?        standby-state
        +--ro sensor-data {hardware-sensor}?
           +--ro value?               sensor-value
           +--ro value-type?          sensor-value-type
           +--ro value-scale?         sensor-value-scale
           +--ro value-precision?     sensor-value-precision
           +--ro oper-status?         sensor-status
           +--ro units-display?       string
           +--ro value-timestamp?     yang:date-and-time
           +--ro value-update-rate?   uint32
```

With IETF YANG-Push we like to subscribe

with on-change sync-on-start

- For inventory use cases the state of the hardware.

with on-change

- For alert use cases the hardware state notifications.

and periodically

- For performance measurement use cases the hardware sensor-data.

```
  notifications:
    +---n hardware-state-change
    +---n hardware-state-oper-enabled {hardware-state}?
    |  +--ro name?          -> /hardware/component/name
    |  +--ro admin-state?   -> /hardware/component/state/admin-state
    |  +--ro alarm-state?   -> /hardware/component/state/alarm-state
    +---n hardware-state-oper-disabled {hardware-state}?
       +--ro name?          -> /hardware/component/name
       +--ro admin-state?   -> /hardware/component/state/admin-state
       +--ro alarm-state?   -> /hardware/component/state/alarm-state
```

Operational IETF/IEEE YANG

IETF 125 Module Implementation Status

	6WIND VSR	Huawei NE	Huawei MA	Cisco IOS XR	Arrcus Arcos
ietf-interfaces.yang (inventory, state and statistic)	✓	P	✓		
ietf-hardware.yang (inventory, state and statistic)	✓		✓		
ietf-alarms.yang (state)	✓		✓		
ieee802-dot1ab-lldp.yang (inventory and state)					
ieee802-dot1ax.yang (inventory, state and statistic)					
ietf-bfd-ip-sh.yang (state)					
ietf-bfd-ip-mh.yang (state)					
ietf-bfd-lag.yang (state)					
ietf-isis.yang (state)					

Green marked describes new capability at IETF 125. "P" to partially implemented.

YANG Message Broker

IETF 125 Implementation Status

	NetGauze	Pmacct	Apache Kafka	Ciena Blueplanet	Cisco Crosswork
draft-ietf-nmop-message-broker-telemetry-message	✓	✓			
YANG-Push Receiver	✓	✓			
YANG Schema Retrieval	✓				
YANG Message Broker Producer	✓				
YANG Schema Registry			✓		
YANG Message Broker Consumer	✓			✓	✓
Validate Telemetry Message against YANG Schema	✓			✓	✓
draft-ietf-netmod-yang-anydata-validation	✓				
YANG Schema Comparison			✓		

[NetGauze](#)
[Pmacct](#)

Green marked describes new capability at IETF 125. "P" to partially implemented.

Problem Statement and Motivation

How it is being addressed in which document

Network Anomaly Detection

When operational or configurational changes in connectivity services are happening, **the objective is to detect interruption at network operation faster than the users using those connectivity services**.

In order to achieve this objective, **automation in network monitoring is required.** This automation needs to **monitor network changes holistically** by monitoring all 3 network planes simultaneously and detect whether that change is service disruptive.

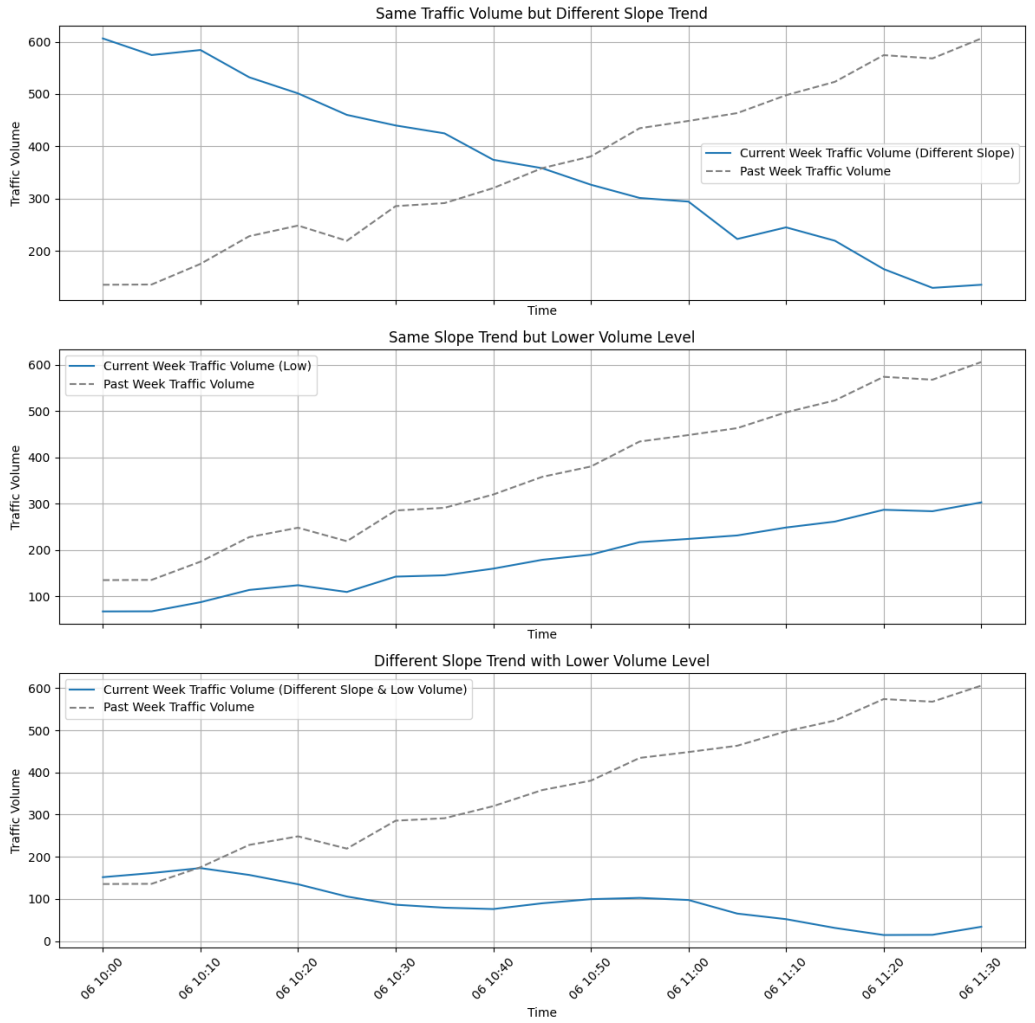
Through network incidents postmortems we network operators **learn and improve so does network anomaly detection** and supervised and semi-supervised machine learning. **With more and more incidents the postmortem process demands automation** and with the standardization of labeled network incident collaboration among network operators, vendors and academia is facilitated.

- draft-ietf-nmop-network-anomaly-architecture describes the motivation and architecture and the relationship to other two documents.
- draft-ietf-nmop-network-anomaly-semantics defines Symptom semantics to enable standardized data exchange to validate results with network engineers and improve supervised and semi-supervised machine learning systems.
- draft-ietf-nmop-network-anomaly-lifecycle describes on managing the lifecycle process, in order to facilitate network engineers to interact with the network anomaly detection system to refine the detection abilities over time.

Cosmos Bright Lights Rules

Missing Traffic

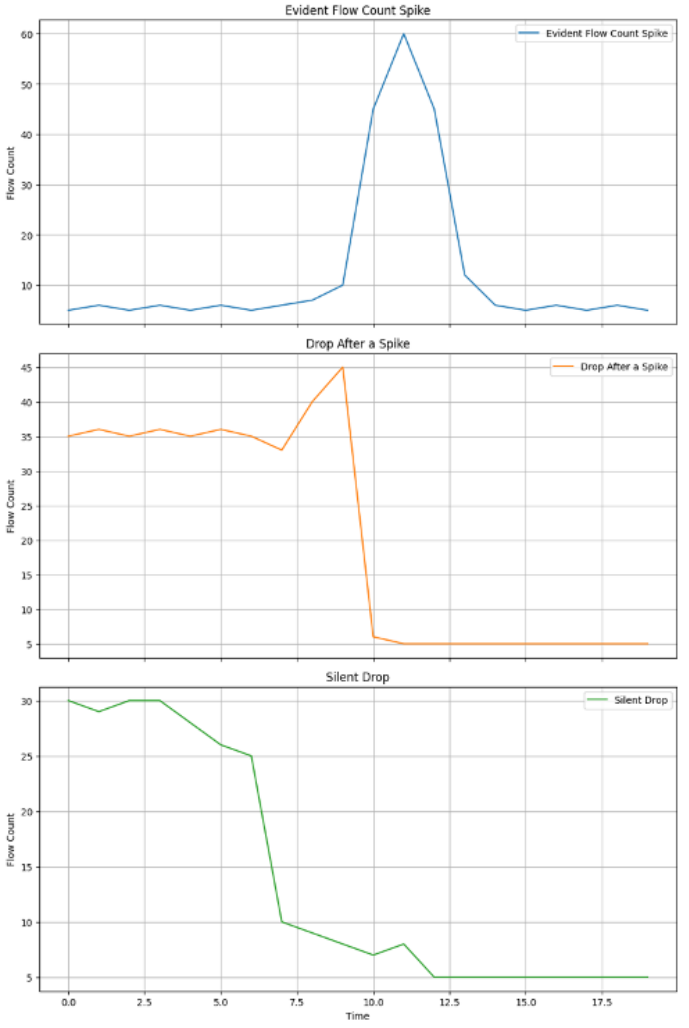
- **Input Metrics**: all IPFIX Bytes within an LC
- **Plane**: Data plane
- **Aim**: Detect traffic drops outside the monitoring domain
- **Concern Object**: None
- **Alerting Aspects**:
  - Traffic volume level
  - Traffic slope trend
- **Alerting Scenarios**:
  - Same traffic volume but different slope trend
  - Same slope trend but a significantly lower volume level
  - Different slope trend and a significantly lower volume level



Cosmos Bright Lights Rules

Sudden Flow Count Change

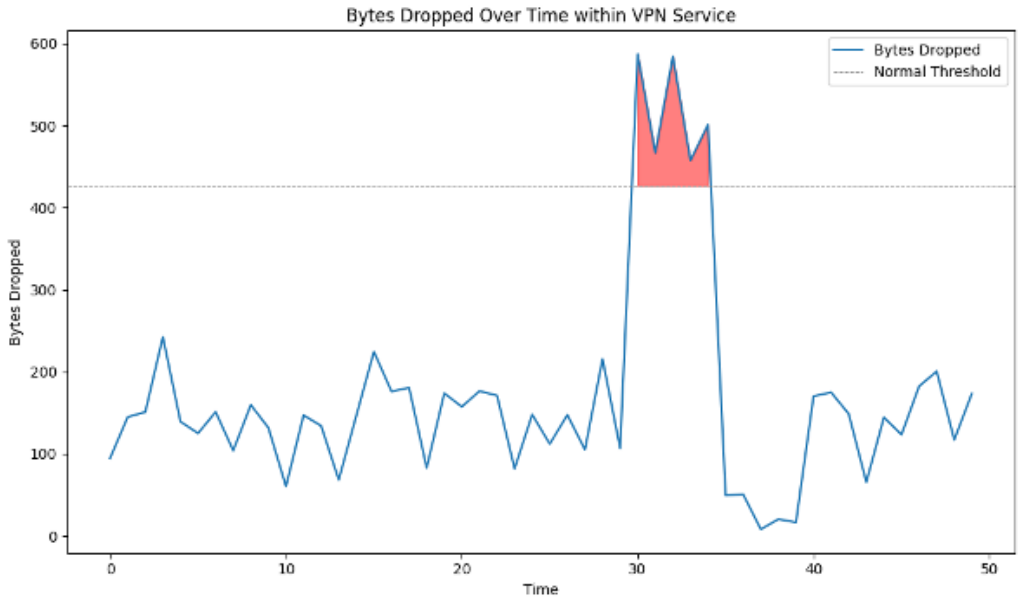
- **Input Metrics**: all IPFIX traffic flow (IE3 deltaFlowCount) within an LC
- **Plane**: Data plane
- **Aim**: Detect Network connections disruption
- **Concern Object**: None
- **Alerting Aspects**:
  - Flow count pattern sudden change
- **Alerting Scenarios**:
  - Sessions re-establishment after interruption
    - **Evident flow count spike**: A sudden and significant increase in flow count.
    - **Drop after a spike**: A flow count spike following a rapid decline in the flow count level
  - Data forwarding is interrupted without being able to re-establish sessions
    - **Silent Drop**: A sudden and significant drop in flow count level without any prior increase or spike.



Cosmos Bright Lights Rules

Dropped Bytes

- **Input Metrics**: IPFIX Bytes within an LC with the following bad IE89 forwardingStatus Drop
  - **Adjacency**
  - **Unrouteable**
  - **Bad TTL**
- **Plane**: Data plane
- **Aim**: Detect excessive network traffic dropping due to network malfunction
- **Concern Object**: (node_id, iface_in, mpls_vpn_rd)
- **Alerting Aspects**:
  - Deviation of the amount of dropped bytes compared to its usual upper bound



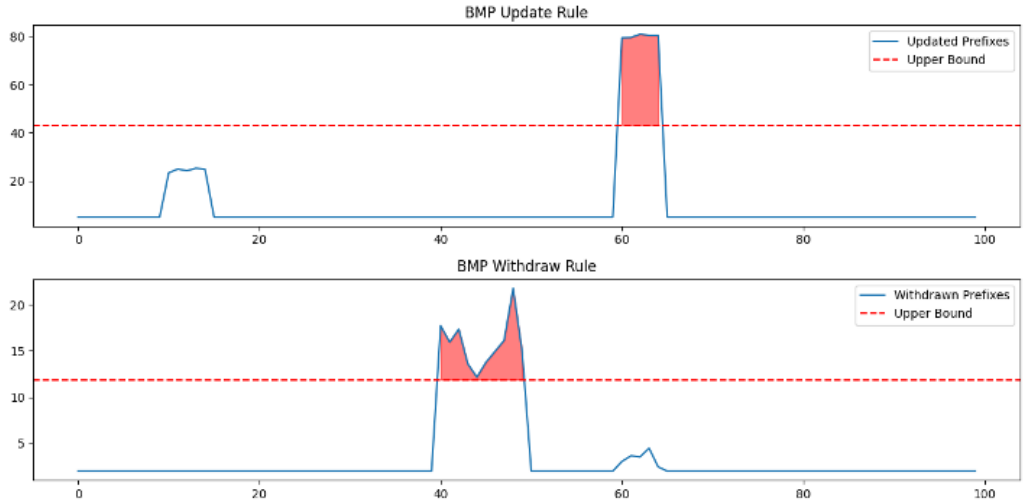
Detected Dropping node-iface-rd

Forwarding Node Id, Iface In, Mpls \	Count ↓
ipd-lss690-r-pe-20	20
1012	20
0:6837:4004710020	20

Cosmos Bright Lights Rules

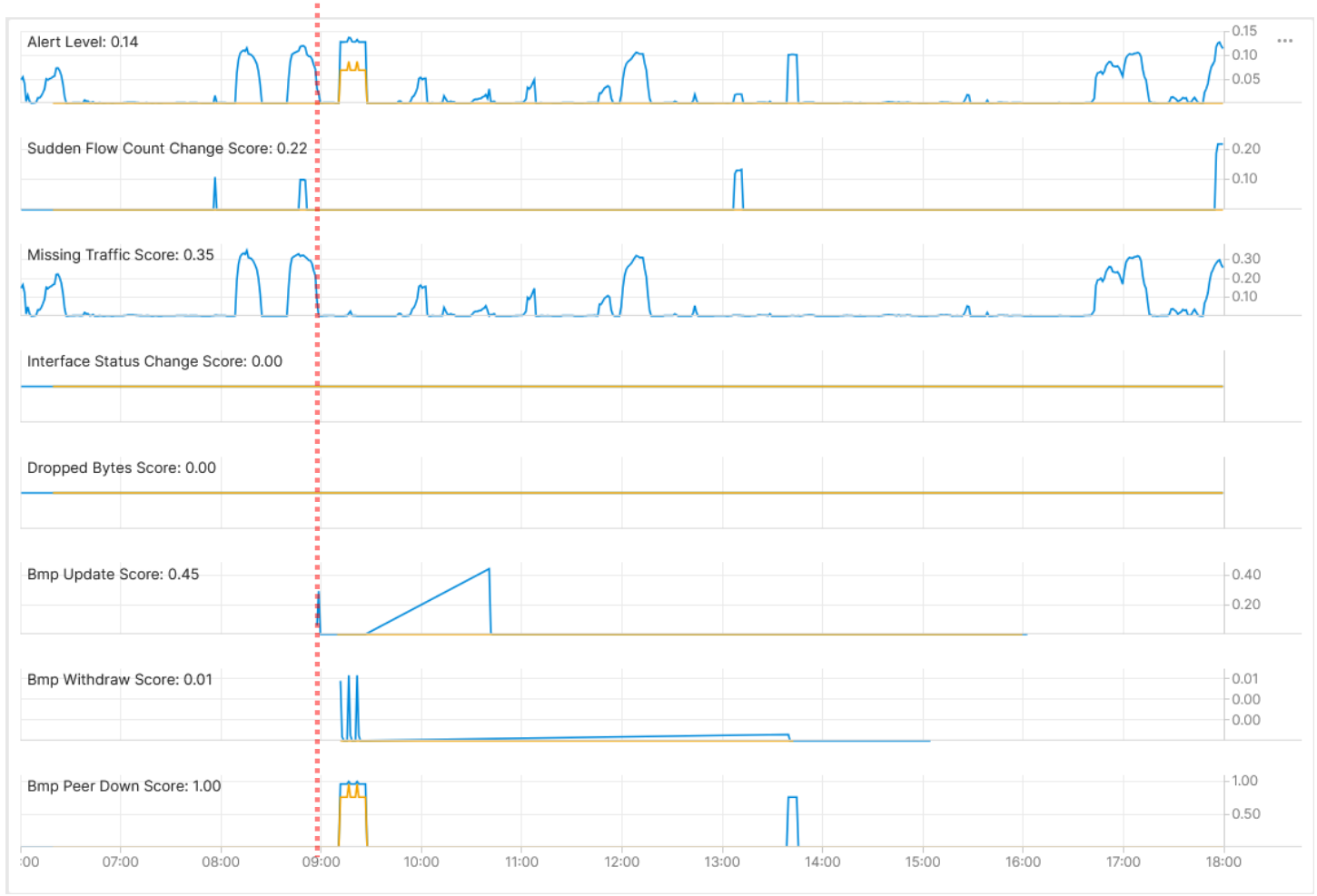
BMP Update and BMP Withdraw

- **Input Metrics**: The total number of prefixes in BMP route monitoring messages within an LC
  with log_type =
  - **update**
  - **withdraw**
- **Plane**: Control plane
- **Aim**:
  Detect abnormal surges in the number of updated or withdrawn prefixes that may indicate unusual or potentially harmful control plane activity.
- **Concern Object**: (next hop, rd)*
  * will be available in the future
- **Alerting Aspects**:
  - Deviation of the number of updated/withdrawn prefixes from the usual upper bounds



September 4th, Maximum Prefix BGP Peer State Change

External Cloud Access, AWS Peering



Cosmos Bright Lights monitoring 64498:46982 64498:55780 L3 VPN in real-time during incident window – [Pivot Link]

Long time ago, when a new L3 VPN connectivity service was created, the maximum prefix limit on a BGP peering at AWS public cloud was defined but not proactively monitored.

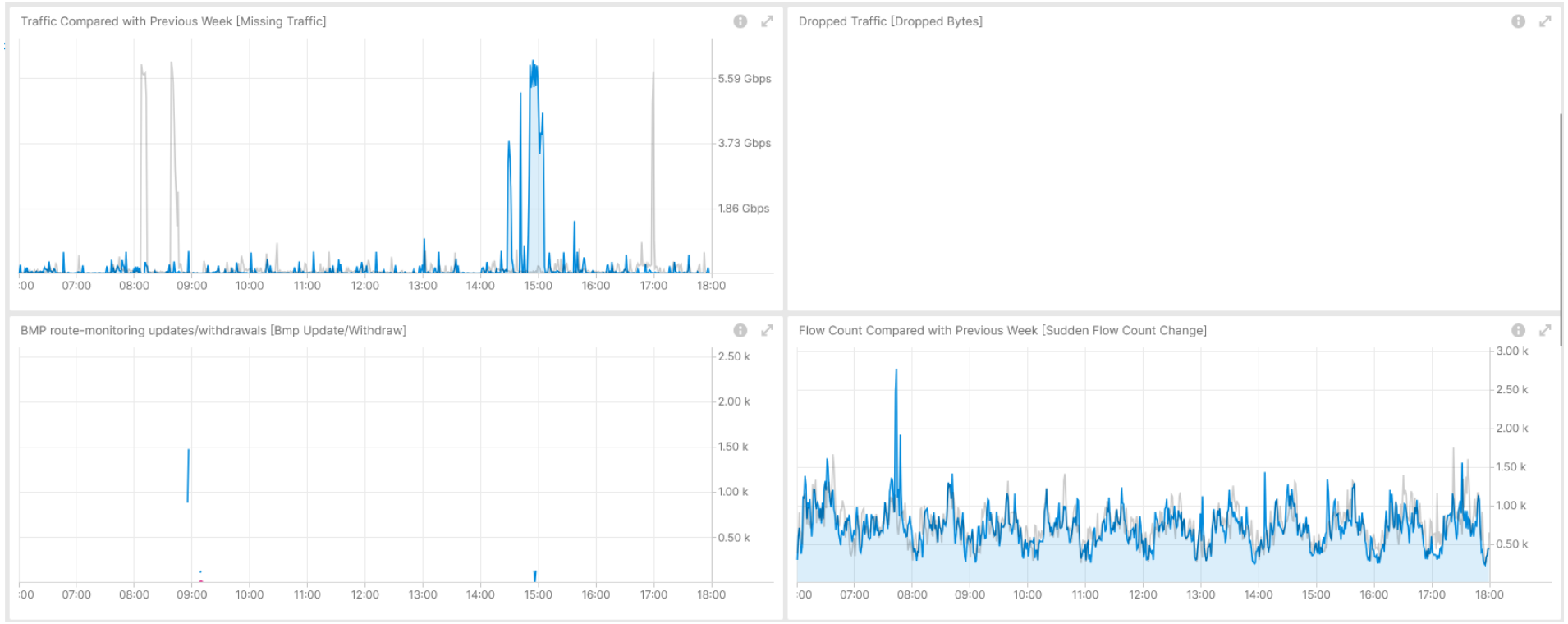
On September 4th at 08:55 additional prefixes were advertised into the L3 VPN and observed by Cosmos Bright Lights Network Anomaly Detection but nor alerted.

At 09:08 additional prefixes were advertised and soon after the BGP peering was teared. Before peering teardown, the AWS BGP speaker notified Swisscom's PE node the reason of the peer down. The Swisscom PE node forwarded in the BMP peer_down notification to the data collection where the peer_down but not the reason was decoded. **Cosmos Bright Lights Network Anomaly Detection observed the topology and peering state changes but did not alert.**

Some time later network operation was contacted due to customer facing incident…

September 4th, Maximum Prefix BGP Peer State Change

64498:46982 and 64498:55780 L3 VPN's – **Real-Time** Incident Analysis



Shows no drops and traffic volume changes, Measured with IPFIX and Correlated with with BGP VPNv4/6, BMP Adj-RIB In and Local RIB.

Shows changes in BGP topology but no changes flow count. Measured with IPFIX and Correlated with BGP VPNv4/6, BMP Adj-RIB In and Local RIB.

Operational Network Telemetry forwarding plane, IPFIX, BMP measured control plane metrics. – Pivot Link

September 4th, Maximum Prefix BGP Peer State Change

BGP notifications and BMP Peer Down Message

RFC 7854 - BGP Monitoring Pro

datatracker.ietf.org/doc/

RFC 7854 **Proposed Standard**

4.9. Peer Down Notification

This message is used to indicate that a peering session was terminated.

```
   0                   1                   2                   3
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
  +-+-+-+-+-+-+-+-+
  |    Reason     |
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
  |            Data (present if Reason = 1, 2 or 3)               |
  ~                                                               ~
  +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Reason indicates why the session was closed. Defined values are:

- o Reason 1: The local system closed the session. Following the Reason is a BGP PDU containing a BGP NOTIFICATION message that would have been sent to the peer.
- o Reason 2: The local system closed the session. No notification message was sent. Following the reason code is a 2-byte field containing the code corresponding to the Finite State Machine (FSM) Event that caused the system to close the session (see Section 8.1 of [RFC4271]). Two bytes both set to 0 are used to indicate that no relevant Event code is defined.
- o Reason 3: The remote system closed the session with a notification message. Following the Reason is a BGP PDU containing the BGP NOTIFICATION message as received from the peer.
- o Reason 4: The remote system closed the session without a notification message. This includes any unexpected termination of the transport session, so in some cases both the local and remote systems might consider this to apply.
- o Reason 5: Information for this peer will no longer be sent to the monitoring station for configuration reasons. This does not, strictly speaking, indicate that the peer has gone down, but it does indicate that the monitoring station will not receive updates for the peer.

RFC 4486: Subcodes for BGP Ce

3. Subcode Definition

The following subcodes are defined for the Cease NOTIFICATION message:

Subcode	Symbolic Name
1	Maximum Number of Prefixes Reached
2	Administrative Shutdown
3	Peer De-configured
4	Administrative Reset
5	Connection Rejected
6	Other Configuration Change
7	Connection Collision Resolution
8	Out of Resources

NetGauze/crates/bgp-pkt at ma

github.com/NetGauze/Ne

Files | main | NetGauze / crates / bgp-pkt / | Top

Supported BGP Protocol features

Supported message types

Message Type	RFCs	notes
Open	RFC 4271	See below for the supported capabilities
Update	RFC 4271	See below for the supported path attributes
Notification	RFC 4271	See below for the supported notif sub-codes
KeepAlive	RFC 4271	
RouteRefresh	RFC 2918 and RFC 7313	See below for the supported Route refresh ops

BGP notifications propagate reason for BGP peer tear down. BMP mirrors BGP notification at BGP Adj-RIB In with reason code 3. If no notification was received but TCP reset was received, then reason code 4 is used.

Show BGP neighbor output in CLI

```
RP/0/RP0/CPU0:pca01ro1072olt#sho bgp vrf EXTCC-00508-00
neighbor 100.100.98.209
.......
Time since last notification received from neighbor:
00:34:13
  Error Code: maximum number of prefixes reached
  Notification data received:
```

September 4th, Maximum Prefix BGP Peer State Change

draft-ietf-nmop-network-anomaly-semantics - Symptoms: **Action – Reason – Trigger**

draft-ietf-nmop-network-anomaly-semantics-03 None

Reachability	Withdraw	Stale
Reachability	Withdraw	Route Policy Filtered
Reachability	Withdraw	Maximum Number of Prefixes Reached
Adjacency	Established	Peer
Adjacency	Established	Link-Layer
Adjacency	Locally Teared Down	Peer
Adjacency	Remotely Teared Down	Peer
Adjacency	Locally Teared Down	Link-Layer
Adjacency	Remotely Teared Down	Link-Layer
Adjacency	Locally Teared Down	Administrative
Adjacency	Remotely Teared Down	Administrative
Adjacency	Locally Teared Down	Maximum Number of Prefixes Reached
Adjacency	Remotely Teared Down	Maximum Number of Prefixes Reached
Adjacency	Locally Teared Down	Transport Connection Failed

https://datatracker.ietf.org/doc/html/draft-ietf-nmop-network-anomaly-semantics-03#symptom_control_plane_actions_table

draft-ietf-nmop-network-anomaly-semantics-03 None

4.2.1. YANG Tree

Figure 1 contains the YANG tree diagram [RFC8340] of the 'ietf-network-anomaly-symptom-cbl' module. It augments the 'ietf-relevant-state' module defined in [I-D.ietf-nmop-network-anomaly-lifecycle].

For each Symptom, the following parameters can be assigned: an Action, a Reason and a Trigger describing the Symptom; a concern score indicating how critical the Symptom is; and the associated network plane.

Where the season enumeration declares wherever a workday or a holiday hase been taken into consideration for Contextual Outliers. The template describes which approach and parameters have been used in the Service Disruption Detection as described in Section 3.2 of [I-D.ietf-nmop-network-anomaly-architecture]

```
module: ietf-network-anomaly-symptom-cbl

  augment /rsn:relevant-state/rsn:anomaly/rsn:symptom:
    +--rw action?          string
    +--rw reason?          string
    +--rw trigger?         string
    +--rw network-plane?   enumeration
    +--rw template?        string
    +--rw season?          enumeration
  augment /rsn:relevant-state-notification/rsn:anomaly/rsn:symptom:
    +-- action?          string
    +-- reason?          string
    +-- trigger?         string
    +-- network-plane?   enumeration
    +-- template?        string
    +-- season?          enumeration
```

https://datatracker.ietf.org/doc/html/draft-ietf-nmop-network-anomaly-semantics-03#section-4.2.1

In Postmortems, the causality is a key concern. With symptoms we describe the causality tree in a human and machine-readable ontology.

The symptoms is defined in semantic triple. Where "action" is the object, "trigger" the predicate and the reason "subject".

With semantic triple we knowledge graph.

IETF NMOP - Semantic Metadata Annotation for Network Anomaly Detection

draft-ietf-nmop-network-anomaly-semantics – Schema Tree

```
notifications:
   +---n relevant-state-notification
      +--ro publisher
      |  +--ro id?        yang:uuid
      |  +--ro name       string
      |  +--ro version?   string
      +--ro id                                yang:uuid
      +--ro uri?                              inet:uri
      +--ro description?                      string
      +--ro start-time                        yang:date-and-time
      +--ro end-time?                         yang:date-and-time
      +--ro smcblsymptom:strategy?            string
      +--ro confidence-score?                 score
      +--ro concern-score                     score
      +--ro (service)?
      |  +--:(smtopology:l2vpn)
      |  |  +--ro smtopology:vpn-service* [vpn-id]
      |  |     +--ro smtopology:vpn-id                string
      |  |     +--ro smtopology:uri?                  inet:uri
      |  |     +--ro smtopology:vpn-name?             string
      |  |     +--ro smtopology:site-ids*             string
      |  |     +--ro smtopology:change-id?            yang:uuid
      |  |     +--ro smtopology:change-start-time?
      |  |     |       yang:date-and-time
      |  |     +--ro smtopology:change-end-time?
      |  |             yang:date-and-time
      |  +--:(smtopology:l3vpn)
      |     +--ro smtopology:vpn-service* [vpn-id]
      |        +--ro smtopology:vpn-id                string
      |        +--ro smtopology:uri?                  inet:uri
      |        +--ro smtopology:vpn-name?             string
      |        +--ro smtopology:site-ids*             string
      |        +--ro smtopology:change-id?            yang:uuid
      |        +--ro smtopology:change-start-time?
      |        |       yang:date-and-time
      |        +--ro smtopology:change-end-time?
      |                yang:date-and-time
```

```
notifications:
   +---n relevant-state-notification
   +--ro anomaly* [id revision]
         +--ro id                                  yang:uuid
         +--ro revision                            yang:counter32
         +--ro uri?                                inet:uri
         +--ro state                               identityref
         +--ro description?                        string
         +--ro start-time
         |       yang:date-and-time
         +--ro end-time?
         |       yang:date-and-time
         +--ro confidence-score?                   score
         +--ro pattern?                            identityref
         +--ro annotator
         |  +--ro id?                yang:uuid
         |  +--ro name               string
         |  +--ro version?           string
         |  +--ro annotator-type?    enumeration
         +--ro symptom!
         |  +--ro id                                yang:uuid
         |  +--ro concern-score                     score
         |  +--ro smcblsymptom:action?              string
         |  +--ro smcblsymptom:reason?              string
         |  +--ro smcblsymptom:trigger?             string
         |  +--ro smcblsymptom:network-plane?       enumeration
         |  +--ro smcblsymptom:template?            string
         |  +--ro smcblsymptom:season?              Enumeration
         +--ro smtopology:vpn-node-terminations*
                  [hostname route-distinguisher]
            +--ro smtopology:hostname                inet:host
            +--ro smtopology:route-distinguisher     string
            +--ro smtopology:peer-ip*                inet:ip-address
            +--ro smtopology:next-hop*               inet:ip-address
            +--ro smtopology:interface-id*           uint32
```

Shows
the observed symptoms, the network dimensions triggering and connectivity service impacted.

Agentic AI and its Network Analytics Applicability

Exampled on Network Anomaly Detection

With postmortems, network operators **gain insights into the reasoning of network incidents** by looking at what has happened, which operational metrics were available, and which analytical conclusions were derived.

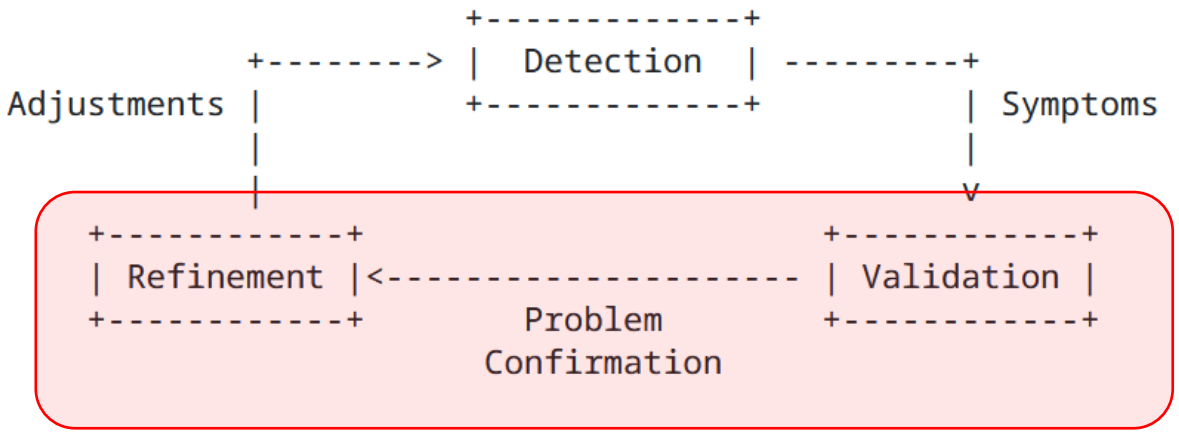
In this process, **additional knowledge is gathered, linked, graphed, and new knowledge derived**.

Without automation this process takes considerate time and human effort. This is where agentic AI comes into play.

"AI Fascinates Throughout History"

Network Anomaly Detection Postmortem Refinement Lifecycle

Gaining **validated** knowledge for **gaining analytical insights**



Use Case 1: Starting from Network Anomaly Detection symptom ontology, **agentic AI assisted context aggregation** from incident tickets, maintenance window story books, postmortem and network platform wikis, can ease human-in-the-loop verification significantly.

Use Case 2: Based on previous validated alerts **agentic AI can pre-label and propose postmortem drafts** on new alerts and thus help network operation not only the information gather but also in the postmortem creation.

Use Case 3: In the refinement phase, **new knowledge-based detection rules and detection strategies or existing refined can be proposed** and through network incident data replay effectiveness can be verified.

Agentic AI and its Network Analytics Applicability
Exampled on **YANG Data Processing Automation**

YANG network data is nested, consists of complex schemas and contains semantic rich information's.

Not all data systems are supporting all YANG data types and nested data structures. Therefore, manual post processing for time series database ingestion is sometimes needed.

When schema validation fails due to software issues in the data acquisition, causality needs to be clarified, and manual network vendor tickets are raised.

Thanks to **schema registry and streaming catalog agentic AI integration**,

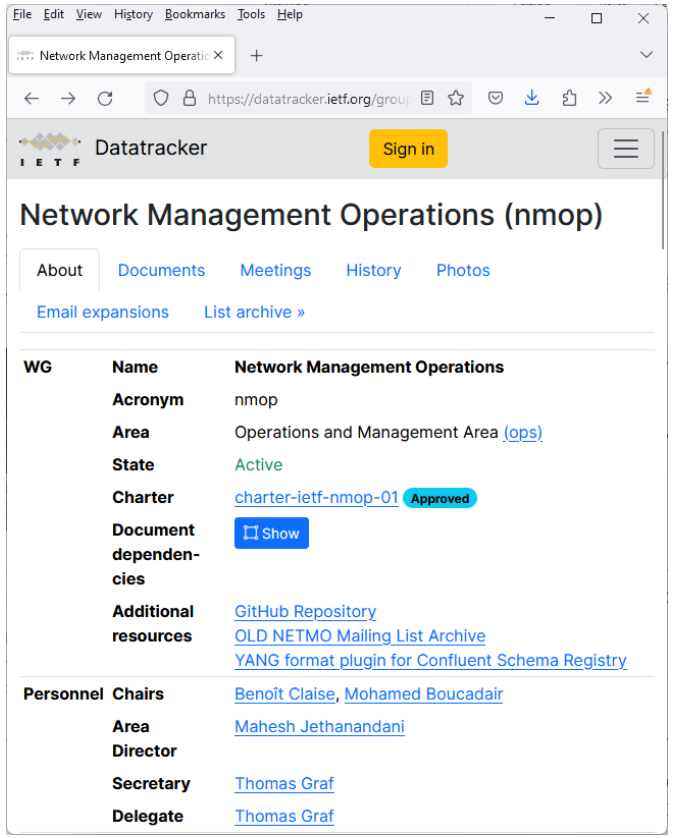
Use Case 4: Agentic AI can **propose YANG transformations** to data stewards.

Use Case 5: Agentic AI can **find schema violation root causes and propose and test YANG schema and data instance fixes** to data owners.

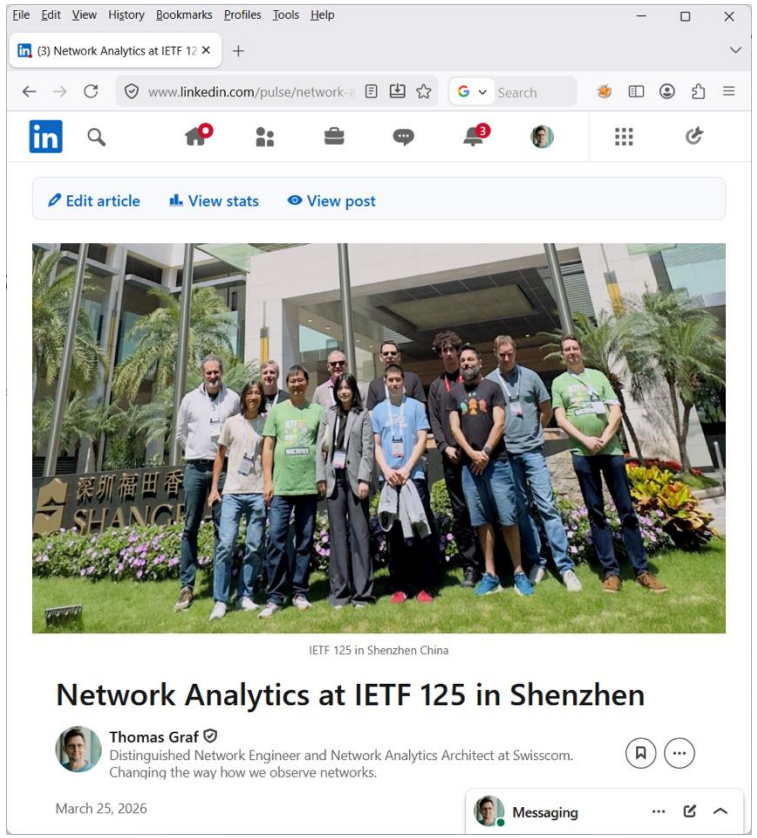
```
module: ietf-interfaces
     +--rw interfaces
        +--rw interface* [name]
           +--rw name                            string
           +--rw description?                    string
           +--rw type                            identityref
           +--rw enabled?                        boolean
           +--rw link-up-down-trap-enable?       enumeration {if-mib}?
           +--ro admin-status                    enumeration {if-mib}?
           +--ro oper-status                     enumeration
           +--ro last-change?                    yang:date-and-time
           +--ro if-index                        int32 {if-mib}?
           +--ro phys-address?                   yang:phys-address
           +--ro higher-layer-if*                interface-ref
           +--ro lower-layer-if*                 interface-ref
           +--ro speed?                          yang:gauge64
           +--ro statistics
              +--ro discontinuity-time    yang:date-and-time
              +--ro in-octets?            yang:counter64
              +--ro in-unicast-pkts?      yang:counter64
              +--ro in-broadcast-pkts?    yang:counter64
              +--ro in-multicast-pkts?    yang:counter64
              +--ro in-discards?          yang:counter32
              +--ro in-errors?            yang:counter32
              +--ro in-unknown-protos?    yang:counter32
              +--ro out-octets?           yang:counter64
              +--ro out-unicast-pkts?     yang:counter64
              +--ro out-broadcast-pkts?   yang:counter64
              +--ro out-multicast-pkts?   yang:counter64
              +--ro out-discards?         yang:counter32
              +--ro out-errors?           yang:counter32
```

IETF NMOP 125/126 – Network Observability Development

Network Anomaly Detection and YANG-Push/Message Broker Integration



https://datatracker.ietf.org/group/nmop/about/



https://www.linkedin.com/pulse/network-analytics-ietf-125-shenzhen-thomas-graf-svmee/.

Next Stop IETF 126 hackathon in Vienna:

Validate YANG-Push to Message Broker End-To-End Data Processing Chain at

validate and verify

- 5 YANG-Push Publishers
- 2 YANG-Push Receivers
- 2 YANG-Push Network Telemetry Message
- 1 YANG Message Broker Producer and Schema Registry
- 3 YANG Message Broker Consumer

implementations in YANG data schema validation and obtaining latest YANG-Push subscription state. Subscribe to YANG data on YANG-Publisher, obtain and register all YANG modules necessary to build YANG schema tree, register YANG schemas to Schema Registry and verify YANG notifications against scheme trees and produce and consume from Message Broker.