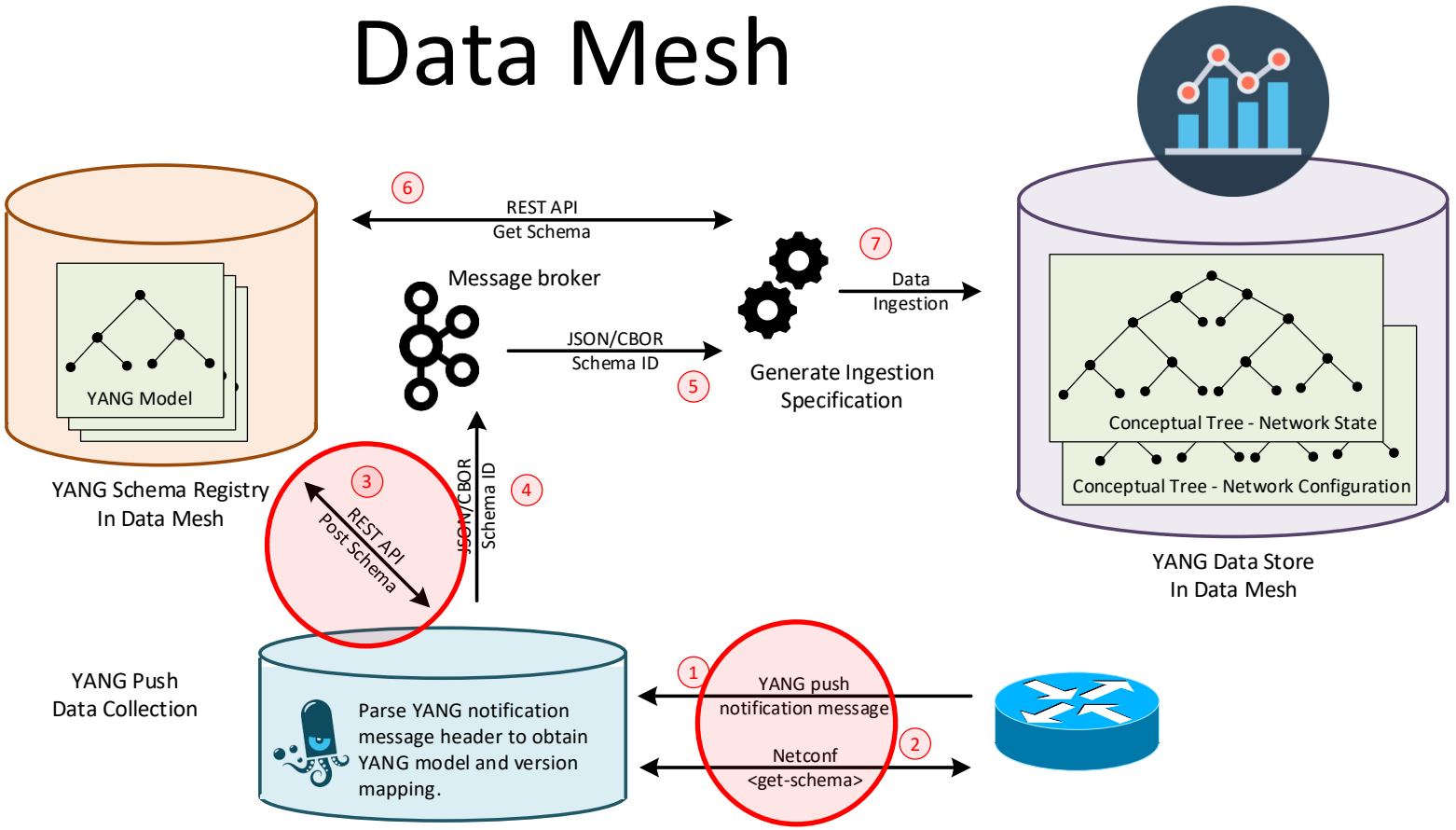
Addressing Network Operator Challenges in YANG push Data Mesh Integration

zhuoyao.lin@huawei.com, jean.quilbeuf@huawei.com
ahmed.elhassany@swisscom.com, alex.huang-feng@insa-lyon.fr
benoit.claise@huawei.com, thomas.graf@swisscom.com

24. July 2023

When Big Data and Network becomes one

Marrying two messaging protocols



- **Data Mesh** is a big data architecture where different domains can exchange data with a **bounded context and SLO's** are defined in Data Products. **Same principle as in networks.**
- **Semantics** are needed to describe the data. **A gauge32 is not the same as counter32.** Values can increase or decrease. Needs monotonic increasing counter normalization or not.
- **Versioning** is needed to not only understand that the semantic has changed, but also wherever the new semantic is backward compatible or not. **Preventing to break the data processing pipeline.**
- **Hostname, publisher ID, sequence numbers and observation timestamping** are needed to **measure loss and delay for SLO's.**
- **YANG push as defined in RFC8641 is missing** hostname, sequence numbers, observation timestamping and versioning**.** draft-ahuang-netconf-notif-yang, draft-tgraf-netconf-notif-sequencing, draft-tgraf-yang-push-observation-time and draft-ietf-netconf-yang-notifications-versioning addresses this.

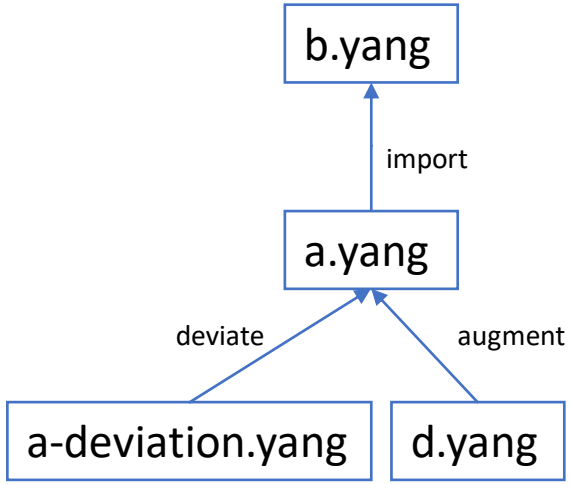
Obtaining YANG semantics and Module dependencies

Overview - Step 1 and 2 in the workflow

Goal: Based on semantic reference in YANG push message, find YANG module dependencies

```
<subscriptions>
  <subscription>
    <id>2222</id>
    <datastore>ds:operational</datastore>
    <datastore-xpath-filter>
       /a-module:a-container
    </datastore-xpath-filter>
    <encoding>encode-xml</encoding>
    <periodic>
      <period>30000</period>
    </periodic>
  </subscription>
</subscriptions>
```

```
module: a-module
 +--rw a-container
   +--rw x?     b:bar
   +--rw d:y
     +--rw d:y-leaf?   b:myenum
```



Subscribe to: **/a-module:a-container**
Require to fetch all these modules.
Then register then into schema registry.

Obtaining the YANG semantics and Module dependencies

Step 1 – Receiving the YANG Push Message

```
jean@jean-vm:~/code/libyangpush/build$ cat push-update.xml
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
  <eventTime>2022-09-02T10:59:55.32Z</eventTime>
  <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
    <id>2222</id>
    <datastore-contents>
      <subscriptions xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notifications">
        <subscription>
          <id>6666</id>
          <datastore xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push"
            xmlns:ds="urn:ietf:params:xml:ns:yang:ietf-datastores">ds:operational</datastore>
          <datastore-xpath-filter xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
              /a-module:a
          </datastore-xpath-filter>
          <transport xmlns:unt="urn:ietf:params:xml:ns:yang:ietf-udp-notif-transport">unt:udp-notif</transport>
          <encoding>encode-xml</encoding>
          <receivers>
            <receiver>
              <name>subscription-specific-receiver-def</name>
              <receiver-instance-ref xmlns="urn:ietf:params:xml:ns:yang:ietf-subscribed-notif-receivers">global-udp-notif-receiver-def</receiver-instance-ref>
            </receiver>
          </receivers>
          <periodic xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
            <period>30000</period>
          </periodic>
        </subscription>
      </subscriptions>
    </datastore-contents>
  </push-update>
</notification>
```

This message is passed to libyangpush. It indicates that a new subscription with 6666 is created and that subscription targets the xpath /a-module:a.

```
jean@jean-vm:~/code/libyangpush/build$ ls ../modules/
jean@jean-vm:~/code/libyangpush/build$
```

Before executing the next step, “modules” folder that caches YANG modules received from the device is empty:

Obtaining the YANG semantics and Module dependencies

Step 2 – Obtaining the YANG semantics

```
jean@jean-vm:~/code/libyangpush/build$ ./demo push-update.xml
response:
{"id":5}
schema id :5
curl_easy_perform() failed: Failed writing received data to disk/application
json dump -1
{"schemaType": "YANG", "references": [], "schema": "module a-module {\n  yang-version 1.1;\n  namespace \"urn:example:yang:a-module\";\n  prefix a;\n\n  container a {\n    list a-instance {\n      key \"name\";\n      leaf n
ame {\n        type string;\n      }\n      leaf state {\n        type string;\n      }\n    }\n  }\n}\n"}

module name: e-module
response:
{"id":6}
schema id :6
curl_easy_perform() failed: Failed writing received data to disk/application
json dump -1
{"schemaType": "YANG", "references": [], "schema": "module e-module {\n  yang-version 1.1;\n  namespace \"urn:example:yang:e-module\";\n  prefix e;\n\n  revision 2023-06-13;\n\n  typedef e-enum {\n    type enumeration {\n
   enum \"zero\";\n    }\n  }\n\n  grouping some-groups {\n    leaf e-leaf {\n      type e-enum;\n    }\n  }\n}\n"}

module name: d-module
response:
{"id":7}
schema id :7
curl_easy_perform() failed: Failed writing received data to disk/application
json dump -1
{"schemaType": "YANG", "references": [{"subject": "hackathon_demo_a-module", "name": "a-module", "version": 1}, {"subject": "hackathon_demo_e-module", "name": "e-module", "version": 1}], "schema": "module d-module {\n  yang-
version 1.1;\n  namespace \"urn:example:yang:d-module\";\n  prefix d;\n\n  import a-module {\n    prefix a;\n  }\n  import e-module {\n    prefix e;\n    revision-date 2023-06-13;\n  }\n\n  augment \"/a:a\" {\n    container
y {\n      leaf y-leaf {\n        type e:e-enum;\n      }\n    }\n  }\n}\n"}
```

```
response:
{"id":8}
schema id :8
curl_easy_perform() failed: Failed writing received data to disk/application
json dump -1
{"schemaType": "YANG", "references": [{"subject": "hackathon_demo_e-module", "name": "e-module", "version": 1}, {"subject": "hackathon_demo_d-module", "name": "d-module", "version": 1}], "schema": "module a-module {\n  yang-
version 1.1;\n  namespace \"urn:example:yang:a-module\";\n  prefix a;\n\n  container a {\n    list a-instance {\n      key \"name\";\n      leaf name {\n        type string;\n      }\n      leaf state {\n        type string;
\n      }\n    }\n  }\n}\n"}
```

The demo fetches all YANG modules from the NETCONF server, parses the input messages and selects the corresponding YANG module needed to build the schema for incoming messages on that subscription to register in the Confluent schema register for serializing the messages in Apache Kafka.

Obtaining the YANG semantics and Module dependencies

Step 2 – Finding the YANG Module dependencies

The modules are registered in the correct order into the YANG schema registry and the corresponding schema-ids are returned. Here is the list of messages for registering the modules:

First module-a (no dependencies)


```
{
  "schemaType": "YANG",
  "references": [],
  "schema": "module a-module {\n  yang-version 1.1;\n  namespace \"urn:example:yang:a-module\";\n  prefix a;\n\n  container a {\n    list a-instance {\n      key \"name\";\n      leaf name {\n        type string;\n      }\n      leaf state {\n        type string;\n      }\n    }\n  }\n}\n"
}
```


Then module e (no dependencies)


```
{
  "schemaType": "YANG",
  "references": [],
  "schema": "module e-module {\n  yang-version 1.1;\n  namespace \"urn:example:yang:e-module\";\n  prefix e;\n\n  revision 2023-06-13;\n\n  typedef e-enum {\n    type enumeration {\n      enum \"zero\";\n    }\n  }\n\n  grouping some-groups {\n    leaf e-leaf {\n      type e-enum;\n    }\n  }\n}\n"
}
```


Then module d (augements a and thus depends on a, depends on e)


```
{
  "schemaType": "YANG",
  "references": [
    {
      "subject": "hackathon_demo_a-module",
      "name": "a-module",
      "version": 1
    },
    {
      "subject": "hackathon_demo_e-module",
      "name": "e-module",
      "version": 1
    }
  ],
  "schema": "module d-module {\n  yang-version 1.1;\n  namespace \"urn:example:yang:d-module\";\n  prefix d;\n\n  import a-module {\n    prefix a;\n  }\n  import e-module {\n    prefix e;\n    revision-date 2023-06-13;\n  }\n\n  augment \"/a:a\" {\n    container y {\n      leaf y-leaf {\n        type e:e-enum;\n      }\n    }\n  }\n}\n"
}
```

Obtaining the YANG semantics and Module dependencies

Step 2 – Register the YANG Semantics in Confluent Schema Registry

Finally, the augmented version of YANG module a, which is the one that matches the passed subscription is registered in the Confluent schema registry. Returning schema id 8 which then can be used to serialize the YANG push push-update message in a Apache Kafka topic.


```
{
  "schemaType": "YANG",
  "references": [
    {
      "subject": "hackathon_demo_e-module",
      "name": "e-module",
      "version": 1
    },
    {
      "subject": "hackathon_demo_d-module",
      "name": "d-module",
      "version": 1
    }
  ],
  "schema": "module a-module {\n  yang-version 1.1;\n  namespace \"urn:example:yang:a-module\";\n  prefix a;\n\n  container a {\n    list a-instance {\n      key \"name\";\n      leaf name {\n        type string;\n      }\n      leaf state {\n        type string;\n      }\n    }\n  }\n}\n"
}
```

Obtaining the YANG semantics and Module dependencies

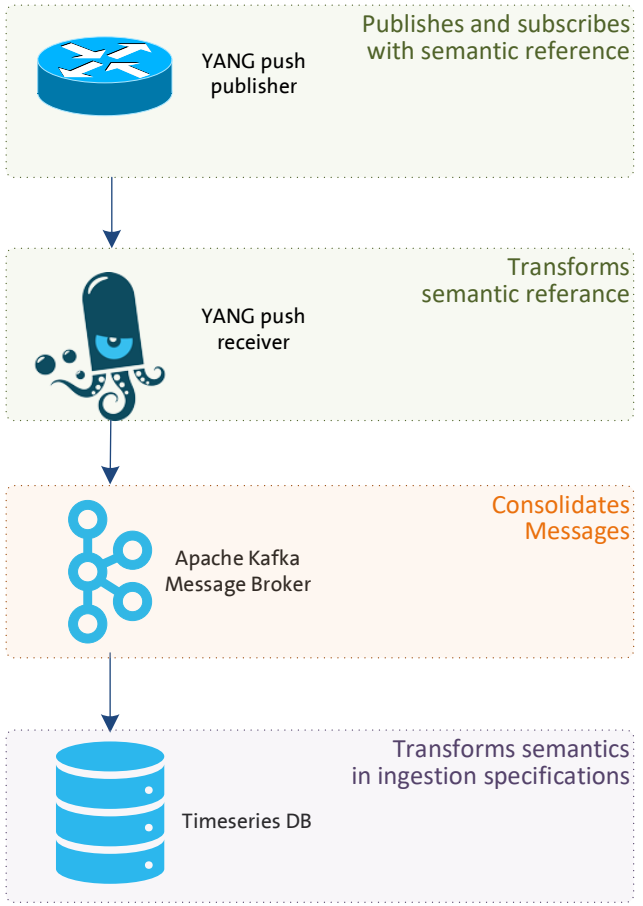
Step 2 – Caching YANG Modules

After the run, the YANG modules are cached in the “modules” folder:

```
jean@jean-vm:~/code/libyangpush/build$ ls ../modules/
a-module.yang                       ietf-keystore@2019-07-02.yang             ietf-origin@2018-02-14.yang                 ietf-tls-common@2019-07-02.yang         sysrepo@2023-02-16.yang
d-module.yang                       ietf-netconf-acm@2018-02-14.yang          ietf-restconf@2017-01-26.yang               ietf-tls-server@2019-07-02.yang         sysrepo-factory-default@2023-02-23.yang
e-module@2023-06-13.yang            ietf-netconf-nmda@2019-01-07.yang         ietf-ssh-common@2019-07-02.yang             ietf-truststore@2019-07-02.yang         sysrepo-monitoring@2022-08-19.yang
iana-crypt-hash@2014-08-06.yang     ietf-netconf-notifications@2012-02-06.yang ietf-ssh-server@2019-07-02.yang             ietf-x509-cert-to-name@2014-12-10.yang  sysrepo-plugind@2022-08-26.yang
ietf-crypto-types@2019-07-02.yang   ietf-netconf-server@2019-07-02.yang       ietf-subscribed-notifications@2019-09-09.yang ietf-yang-patch@2017-02-22.yang
ietf-factory-default@2020-08-31.yang ietf-netconf-with-defaults@2011-06-01.yang ietf-tcp-client@2019-07-02.yang           ietf-yang-push@2019-09-09.yang
ietf-interfaces@2018-02-20.yang     ietf-netconf.yang                         ietf-tcp-common@2019-07-02.yang             nc-notifications@2008-07-14.yang
ietf-ip@2018-02-22.yang             ietf-network-instance@2019-01-21.yang     ietf-tcp-server@2019-07-02.yang             notifications@2008-07-14.yang
```

Obtaining the YANG semantics and Module dependencies

Status and Next Steps



Status

- Parse YANG push subscription state change notification messages for semantic and subscription reference, push-update messages for subscription reference and cache them.
- Determine YANG module dependencies based on YANG library RFC 8525
- Obtain YANG modules and register in Confluent Schema Registry
- Able to compare two YANG module revisions and determine which part of the semantics are not backward compatible

Next Step

- Propose changes in netconf notification header to validate properly in libyang
- Enable schema validation and data serialization in Apache Kafka

zhuoyao.lin@huawei.com, jean.quilbeuf@huawei.com
ahmed.elhassany@swisscom.com, alex.huang-feng@insa-lyon.fr
benoit.claise@huawei.com, thomas.graf@swisscom.com

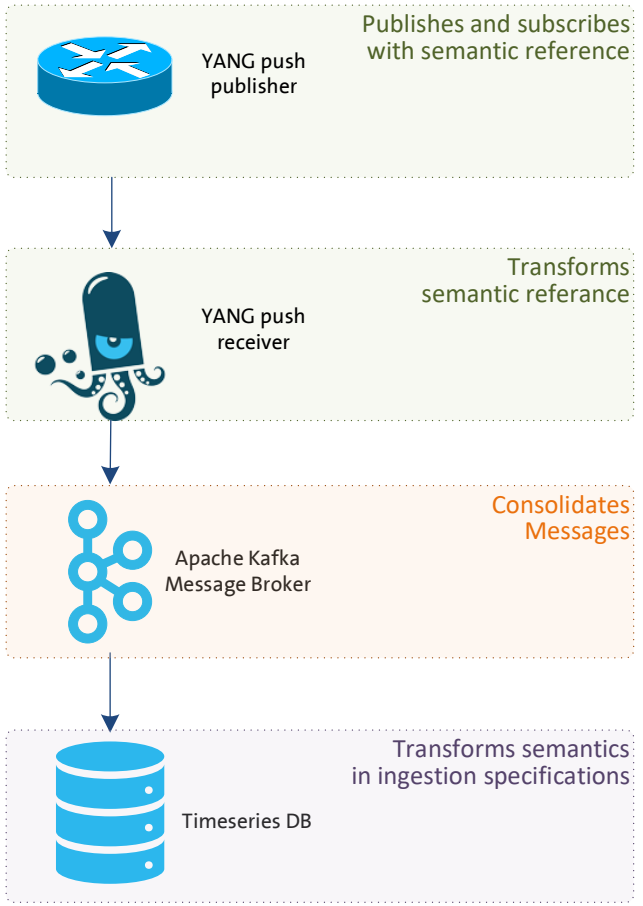
24. July 2023

Netconf Working Group update

Versioning in YANG Notifications Subscription

Status and Next Steps

- **-03 NETCONF adoption call concluded**
- **Feedback from Andy during adoption call**
  - xpath can refer to more than one YANG module
  - subscription section 2 is not fully aligned with section 4.1
  - Instead of sn:target, sn:stream-filter, sn:within-subscription and yp:within-subscription should be augmented
- **All addressed in -01 revision**
  -> Requesting feedback and comments.



thomas.graf@swisscom.com
benoit.claise@huawei.com
alex.huang-feng@insa-lyon.fr

15. July 2023

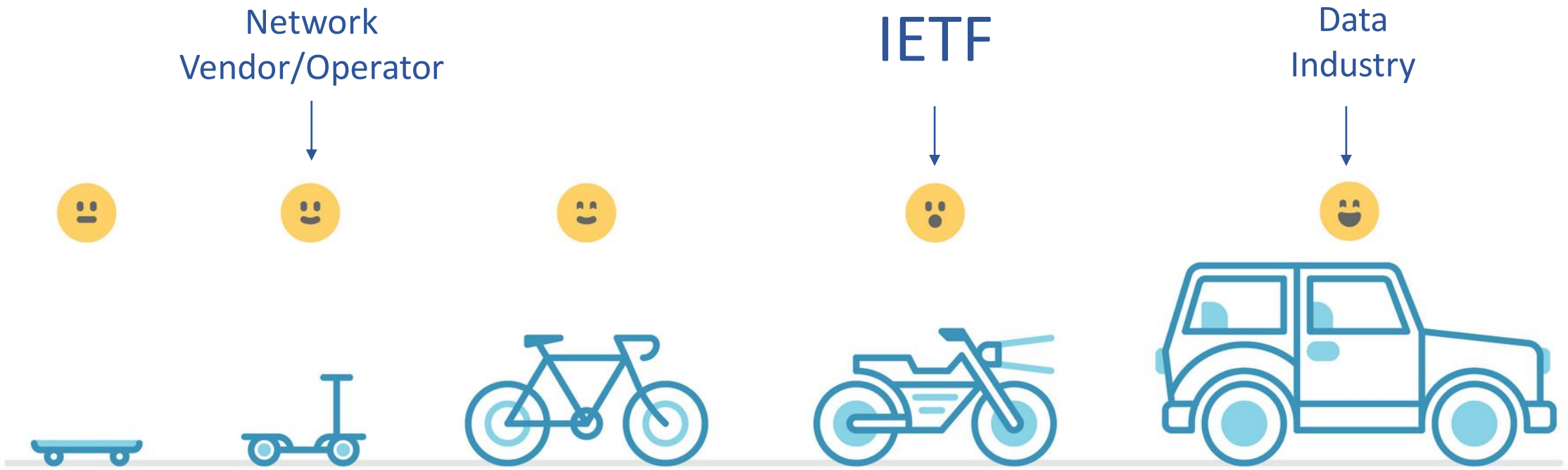
Extend Datastore Selection and Subscription State Change Notifications with module name, revision and revision-label

```
module: ietf-yang-push-revision
     augment /sn:establish-subscription/sn:input/sn:target/sn:stream
               /sn:stream-filter/sn:within-subscription:
       +-- module-version* [module-name]
          +-- module-name       yang:yang-identifier
          +-- revision?         rev:revision-date-or-label
          +-- revision-label?   ysver:version
     augment /sn:establish-subscription/sn:input/sn:target/yp:datastore
               /yp:selection-filter/yp:within-subscription:
       +-- module-version* [module-name]
          +-- module-name       yang:yang-identifier
          +-- revision?         rev:revision-date-or-label
          +-- revision-label?   ysver:version
     augment /sn:subscription-started/sn:target/sn:stream
               /sn:stream-filter/sn:within-subscription:
       +--ro module-version* [module-name]
          +--ro module-name       yang:yang-identifier
          +--ro revision          rev:revision-date-or-label
          +--ro revision-label?   ysver:version
     augment /sn:subscription-started/sn:target/yp:datastore
               /yp:selection-filter/yp:within-subscription:
       +--ro module-version* [module-name]
          +--ro module-name       yang:yang-identifier
          +--ro revision          rev:revision-date-or-label
          +--ro revision-label?   ysver:version
```

```
{
    "ietf-restconf:notification" : {
      "eventTime": "2023-01-03T10:00:00Z",
      "ietf-subscribed-notifications:subscription-modified": {
        "id": 101,
        "stream-xpath-filter": "/ietf-interfaces:interfaces",
        "stream": "NETCONF",
        "ietf-yang-push-revision:module-version": [{
          "module-name": "ietf-interfaces"
          "revision": "2014-05-08",
          "revision-label": "1.0.0",
        }],
      }
    }
  }
```

- **Network operators need to control semantics in its data processing pipeline. That includes YANG push.**
- This is today only possible during YANG push subscription but not when nodes are being upgraded or messages are being published for configured subscription.
- **draft-ietf-netconf-yang-notifications-versioning** extends the YANG push subscription and publishing mechanism defined in RFC8641:
  - **By adding the ability to subscribe to a specific revision** or latest-compatible-semversion of one or more yang modules.
  - **By extending the YANG push Subscription State Change Notifications Message** so that the YANG push receiver learns beside the xpath and the sub-tree filter also the yang module name, revision and revision-label.

Backup



State of the Union
From data mess to data mesh

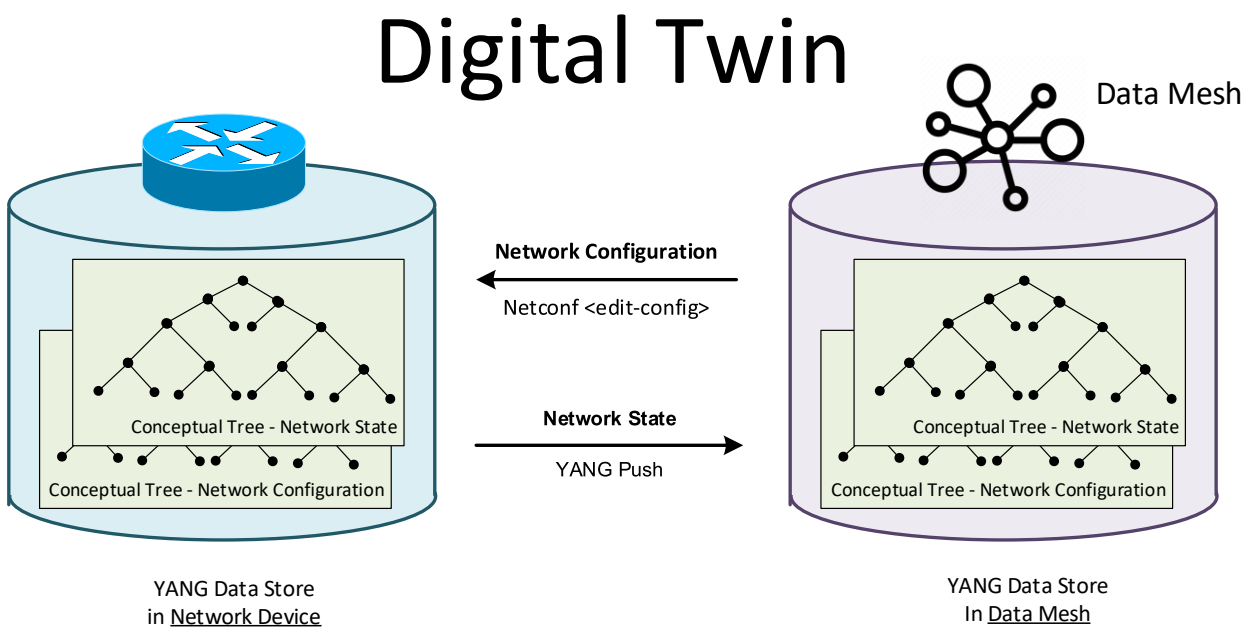
Evolving YANG Push

Missing puzzle pieces

YANG Push	Today at Network Operators	Today at IETF
Transport Protocol	Many and non-standard	netconf-https-notif and netconf-udp-notif
Encoding	JSON widely adopted. Propriety protobuf in various variants. CBOR not implemented yet.	XML in RFC7950, JSON in RFC7951, CBOR in RFC9254
Subscription	Non-standard, periodical widely adopted. On-change sparse.	RFC8639 and RFC8641
Metadata	Non-standard. Partially among message content.	netconf-yang-notifications-versioning, draft-tgraf-netconf-notif-sequencing, draft-tgraf-yang-push-observation-time, draft-claise-opsawg-collected-data-manifest, draft-claise-netconf-metadata-for-collection
Versioning	Neither covered in subscription nor in publishing.	netmod-yang-module-versioning
YANG module	Non-standard widely adopted. IETF coverage non-existent.	Many RFC's defined

YANG datastores enabling Closed Loop Operation

Automated data onboarding with bounded context



YANG is a data modelling language which will not only transform how we managed our networks; it will transform also how we manage our services.

24 industry leading colleagues from 4 network operators, 2 network and 3 analytics providers, and 3 universities **commit on a project to integrate YANG and CBOR into data mesh. IETF 117 public side meeting on Monday July 24th 17:00 – 17:45.**

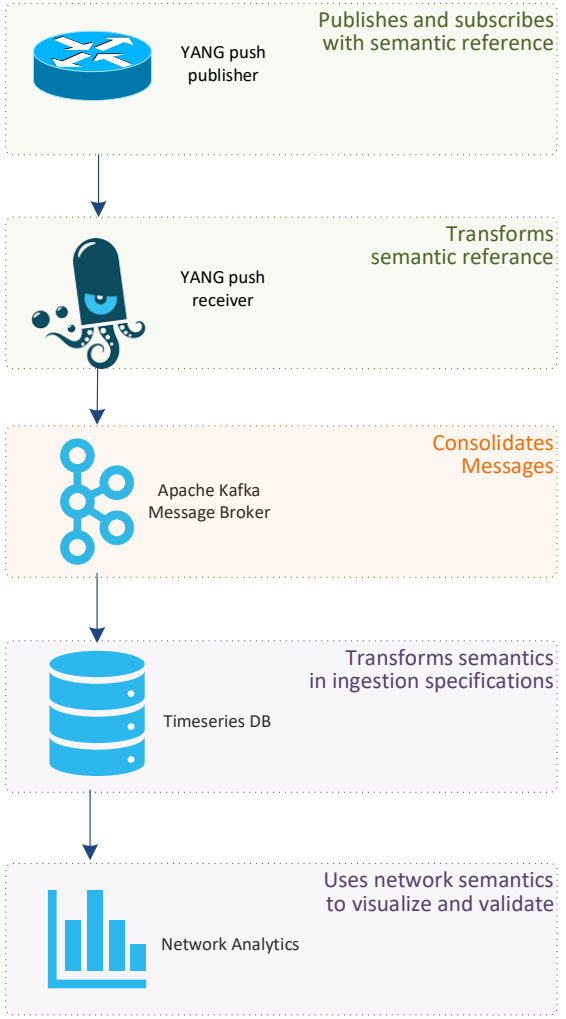
Automated networks can only run with a common data model. A digital twin YANG data store enables a comparison between intent and reality. Schema preservation enables closed loop operation. **Closed Loop is like an autopilot on an airplane.** We need to understand what the flight envelope is to keep the airplane within. Without, we crash.

From YANG push to Analytics

Aiming for an automated 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 **IETF defines the 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.



Evolving Big Data Architecture

Domain oriented, like networks

1st Generation

Proprietary
Enterprise Data Warehouse

2nd Generation

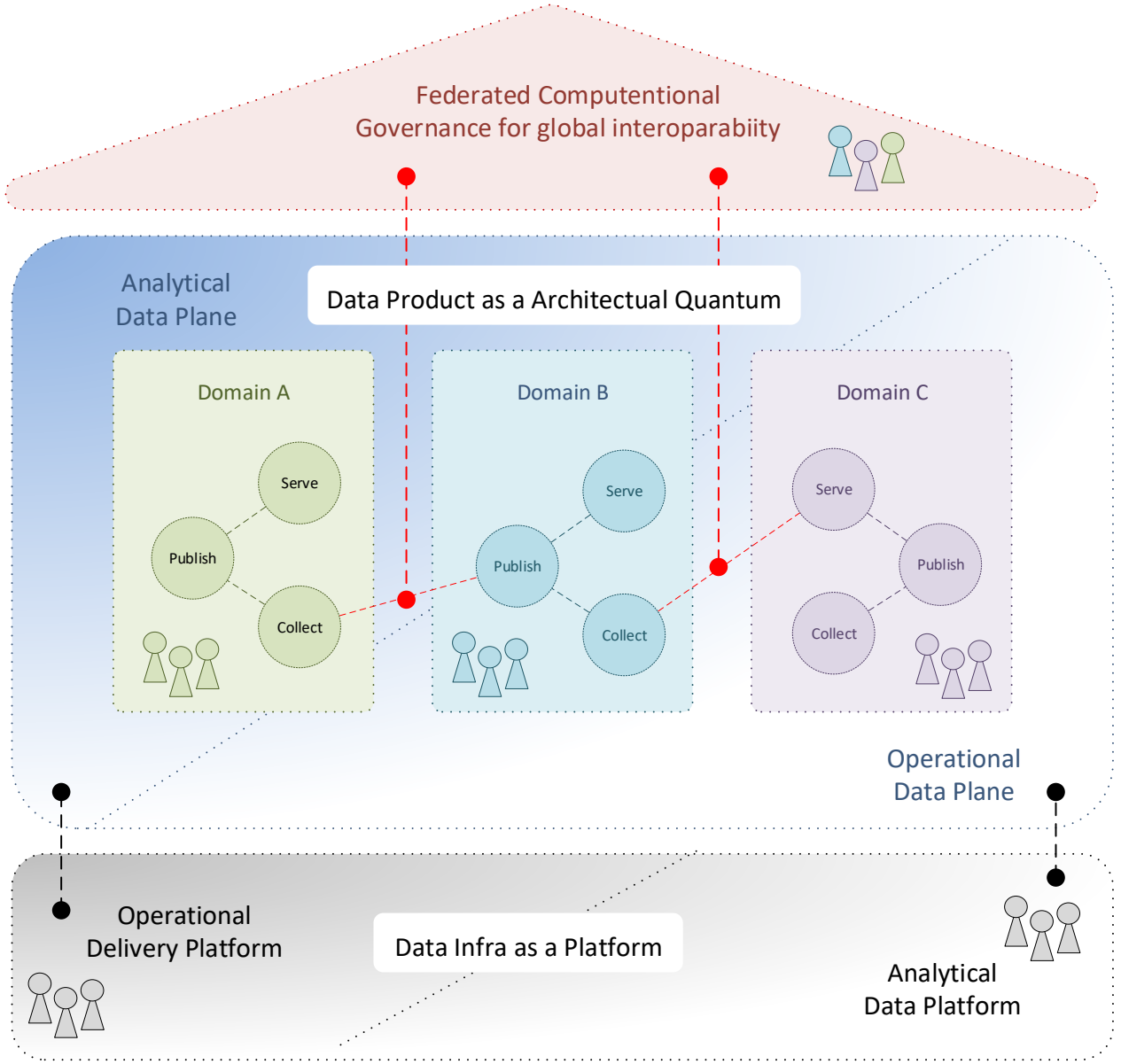
Data lake
Big data ecosystem

3rd Generation current

Kappa
Adds streaming for real-time data

4th Generation next-step

Data Mesh
Distributed and organized in domains.



From Principles to Logical Architecture

From Network to Alert Event

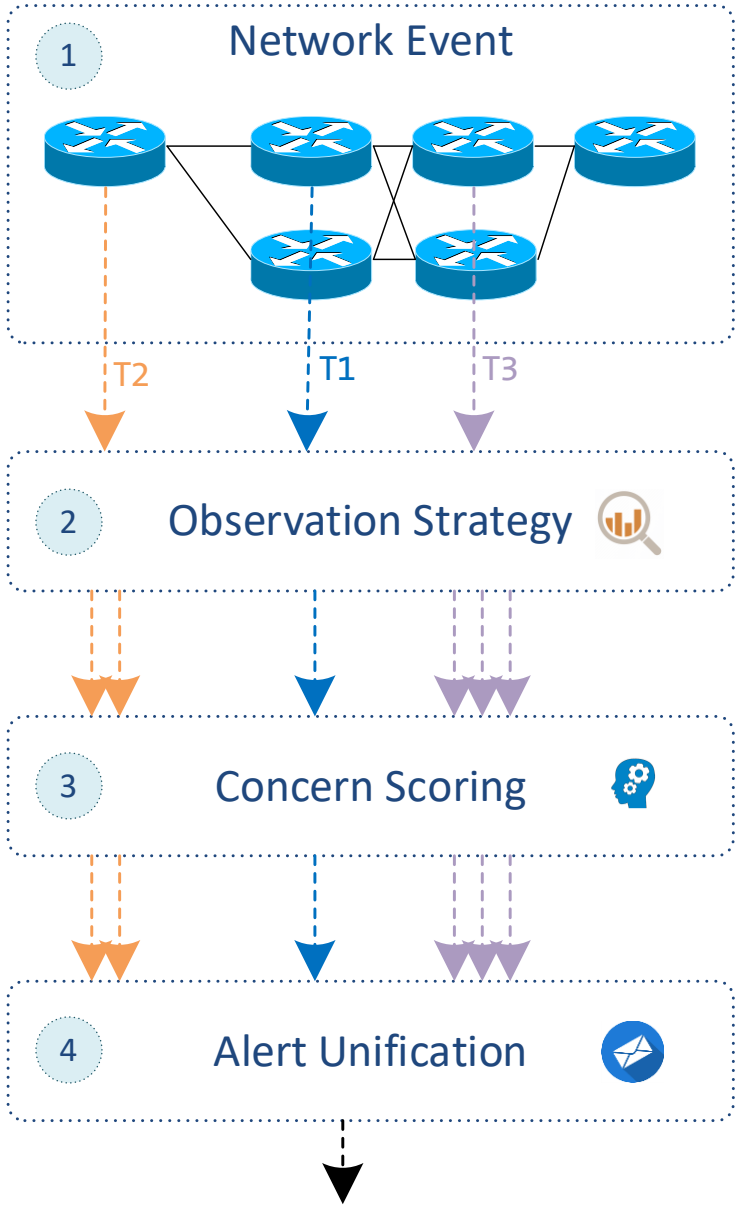
Observe multiple perspectives at different times

1. **A single link down** results in multiple device topology, control-plane and forwarding-plane events being exposed at different times.

2. **Determine** which interfaces and BGP peerings are being used first and then observe state. **Observe** BGP withdrawals and updates, traffic drop spikes and missing traffic. Generate multiple concerns.

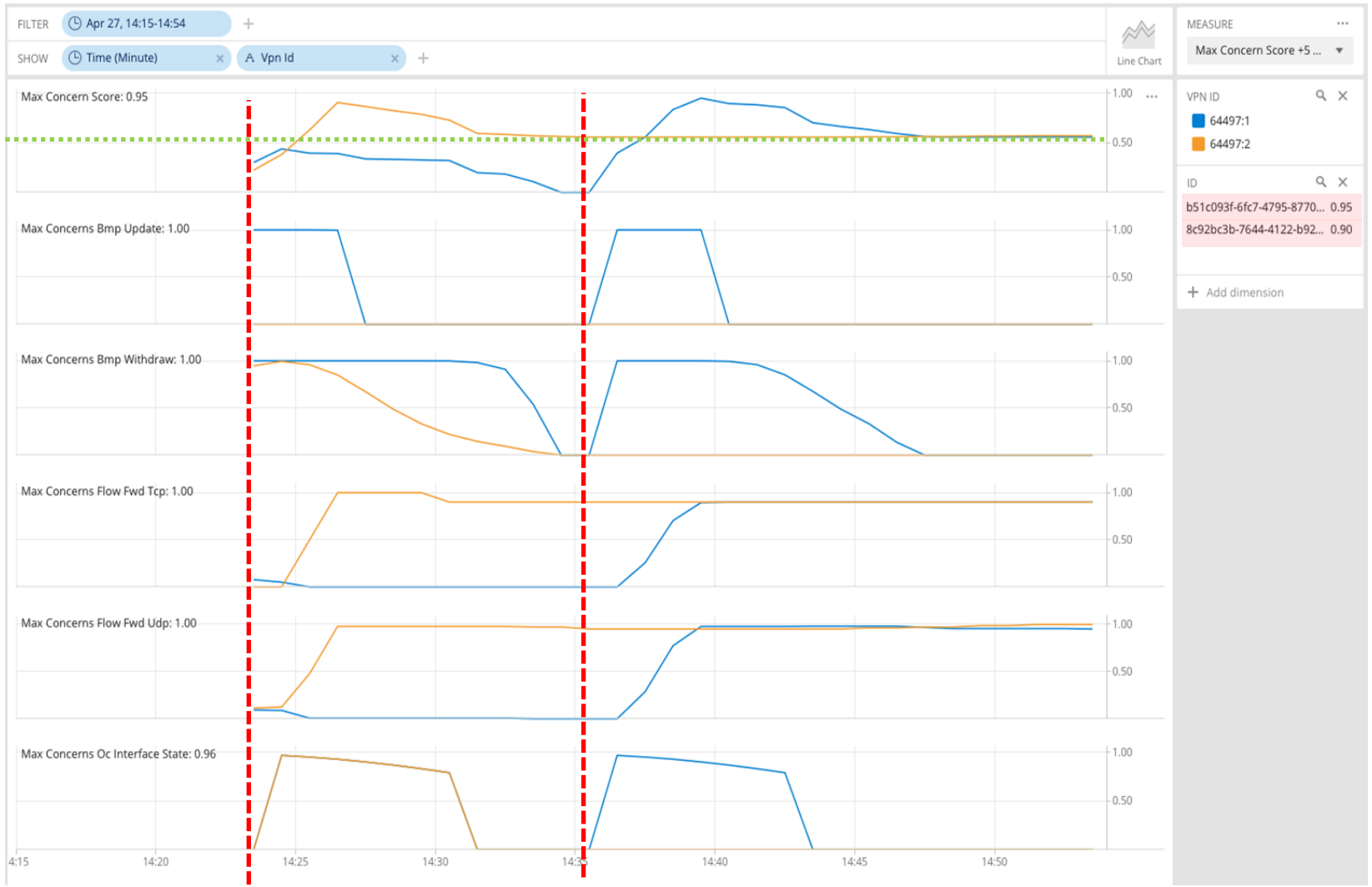
3. **Calculate** for each observation a concern score between 0 and 1. **The higher, the more probable** the changes impacted forwarding.

4. **Unify** several concerns for one VPN connectivity service to one alert identifier.



L3 VPN Network Anomaly Detection

Verify operational changes automatically



Analytical Perspectives

Monitors the network service and wherever it is congested or not.

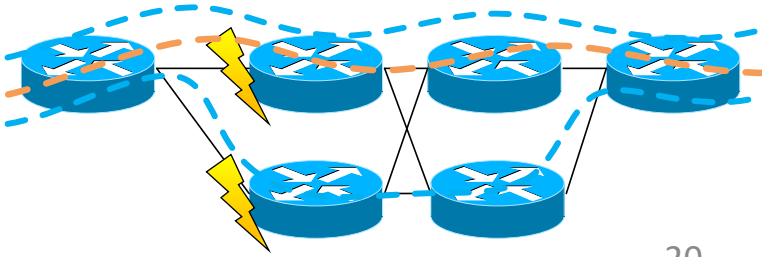
- BGP updates and withdrawals.
- UDP vs. TCP missing traffic.
- Interface state changes.

Network Events

1. VPN orange lost connectivity.
   VPN blue lost redundancy.
2. VPN blue lost connectivity.

Key Point

- AI/ML **requires** network intent and network modelled data to deliver dependable results.



Define YANG module for Netconf Notifications

Closing the semantic gap

```
module: ietf-notification

  structure notification:
    +-- eventTime    yang:date-and-time
```

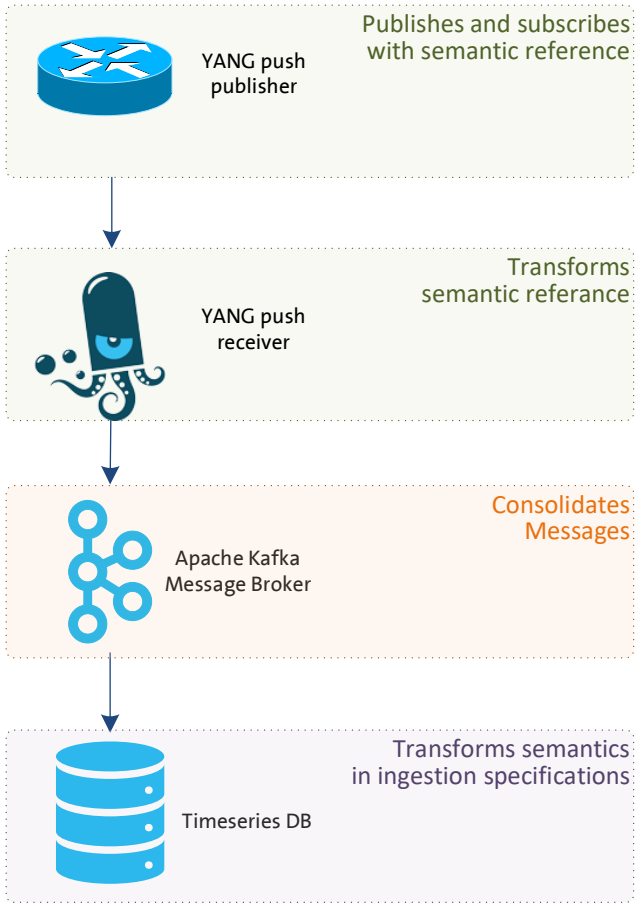
```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>2023-02-04T16:30:11.22Z</eventTime>
<push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
   <id>1011</id>
   <datastore-contents>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-
interfaces">
       <interface>
         <name>eth0</name>
         <oper-status>up</oper-status>
       </interface>
     </interfaces>
   </datastore-contents>
 </push-update>
</notification>
```

- **With RFC 5277 the XML schema for NETCONF event notification was defined.**
- With **draft-ahuang-netconf-notif-yang** updates RFC 5277 by defining the schema as a YANG module.
- **This enables YANG-push to define semantics for the entire YANG push message and use other encodings** than XML such as YANG-JSON RFC 7951 or YANG-CBOR RFC 9264.

Define YANG module for Netconf Notifications

Status

- The yang module prefix has changed to “inotif” to be more explicit.
- The namespace is changed to the one used in RFC5277 : urn:ietf:params:xml:ns:netconf:notification:1.0
- In IANA section, instead of asking for a new URI, we ask IANA to add this document as a reference to the URI from RFC5277
- **Requesting NETCONG working group adoption.**



Extend Streaming Update Notifications with Hostname and Sequencing

For push-update and push-change-update

```
module: ietf-notification-sequencing

  augment-structure /inotif:notification:
    +-- sysName           inet:host
    +-- publisherId       yang:gauge32
    +-- sequenceNumber    yang:counter32
```

```
<notification
xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>2023-02-04T16:30:11.22Z</eventTime>
 <sysName xmlns="urn:ietf:params:xml:ns:yang:ietf-notification-
sequencing">
  example-router
 </sysName>
 <publisherId xmlns="urn:ietf:params:xml:ns:yang:ietf-
notification-sequencing">
  1
</publisherId>
 <sequenceNumber xmlns="urn:ietf:params:xml:ns:yang:ietf-
notification-sequencing">
  187653
 </sequenceNumber>
 <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
   <id>1011</id>
   <datastore-contents>
      <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-
interfaces">
       <interface>
         <name>eth0</name>
         <oper-status>up</oper-status>
       </interface>
     </interfaces>
   </datastore-contents>
 </push-update>
</notification>
```

- When the **NETCONF event notification message is forwarded from the YANG push receiver to another system**, such as a messaging system or a time series database where the message is stored, the **transport context is lost since it is not part of the NETCONF event notification message metadata.** Therefore, the downstream system is unable to associate the message to the publishing process (the exporting router), nor able to detect message loss or reordering.
- **draft-tgraf-netconf-notif-sequencing** extends the NETCONF notification defined in RFC5277 with:
  - **sysName:** Describes the hostname following the 'sysName' object definition in RFC1213 from where the message was published from.
  - **publisherId:** netconf-distributed-notif describes the ability to publish from network processors directly. With this identifier the publishing process from where the message was published from can be uniquely identified.
  - **sequenceNumber:** Generates a unique sequence number as described in RFC9187 for each published message.

Extend Streaming Update Notifications with Observation Timestamping

For push-update and push-change-update

```
module: ietf-yang-push-netobs-timestamping

    augment /yp:push-update:
      +--ro observation-time?   yang:date-and-time
    augment /yp:push-change-update:
      +--ro state-changed-observation-time?   yang:date-and-time
```

```
{
   "ietf-notification:notification": {
      "eventTime": "2023-02-04T16:30:11.22Z",
      "sysName": "example-router",
      "sequenceNumber": 187653,
      "ietf-yang-push:push-update": {
         "id": 1011,
         "observation-time": "2023-02-04T16:30:09.44Z",
         "datastore-xpath-filter": "ietf-interfaces:interfaces",
         "datastore-contents": {
            "ietf-interfaces:interface": {
               "name": {
                  "eth0": {
                     "oper-status": "up"
                  }
               }
            }
         }
      }
   }
}
```

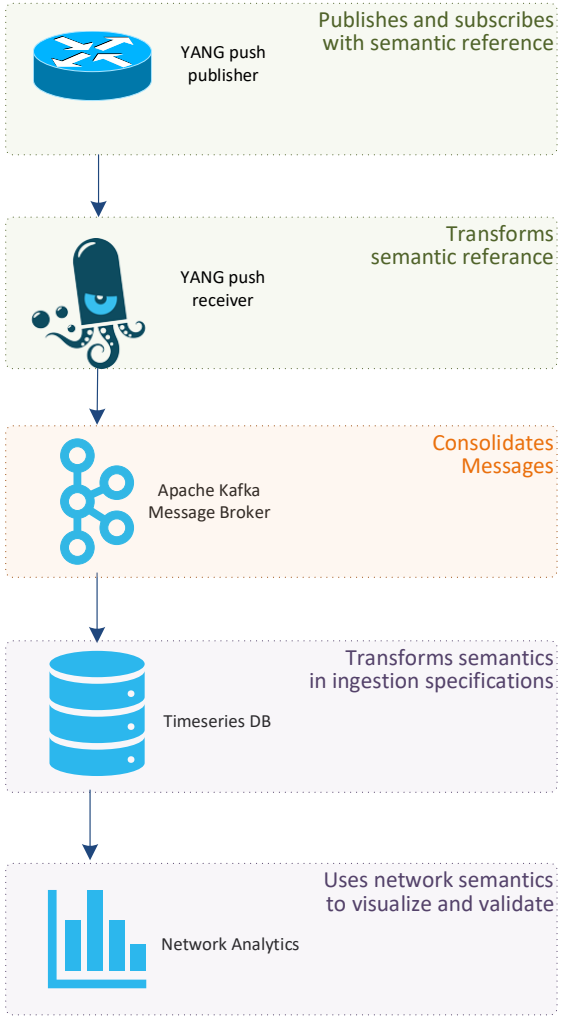
- **To correlate network data** among different Network Telemetry planes as described in Section 3.1 of RFC9232 or among different YANG push subscription types defined in Section 3.1 of RFC8641, **network observation timestamping is needed to understand the timely relationship among these different planes and YANG push subscription types.**
- **draft-tgraf-yang-push-observation-time** extends the YANG push streaming update notification defined in RFC8641 with:
  - **observation-time:** Describes the measurement observation time for the "push-update" notification in a "periodical" subscription.
  - **state-changed-observation-time:** Describes in the "push-change-update" notification in an "on-change" subscription the time when the network state change was observed after the subscription was initially established. In case of an "on-change sync on start" subscription it describes the time when the network state change was observed before the subscription was established.

From YANG push to Analytics

Next steps

- **Do you realize the gaps and how it could be resolved?**
  - By defining a YANG module for NETCONF notification and adding hostname, publisher ID, sequence number, observation time, YANG module name, revision and revision-label into YANG push-update and Subscription State Change notification messages an **automated data processing pipeline** which starts with YANG push, consolidates at Data Mesh and ends at Network Analytics would become at reach.
- **-> What are your thoughts and comments?**
- **-> Interested to learn more? Join the IETF 117 public side meeting on Monday July24th 17:00-17:45 in room Continental 2-3 or look at the project page:**

https://github.com/graf3net/draft-daisy-kafka-yang-integration/blob/main/draft-daisy-kafka-yang-integration-04.md



zhuoyao.lin@huawei.com, jean.quilbeuf@huawei.com
ahmed.elhassany@swisscom.com, alex.huang-feng@insa-lyon.fr
benoit.claise@huawei.com, thomas.graf@swisscom.com