YANG push Integration into Apache Kafka

Zhuoyao Lin

IETF 118 - GROW

TABLE OF CONTENT

I. INTRODUCTION: Motivation

Challenge of Network Management

Networks are growing and being more and more complex to manage.

A costly task for network operators is to identify and correct network issues.

This involves three steps:

- **Detect** that there is an issue,
- **Identify** the cause of issue,
- **Correct** the issue

Today, it is often the customer that notifies the operator about an issue.
Then the cause detection and correction of an issue is done by expert engineers.

The challenge of the network management industry today, is to automate the detection, identification and correction of the problem, which is the frame we are trying to achieved.

The project will be **deployed into production** at the end.

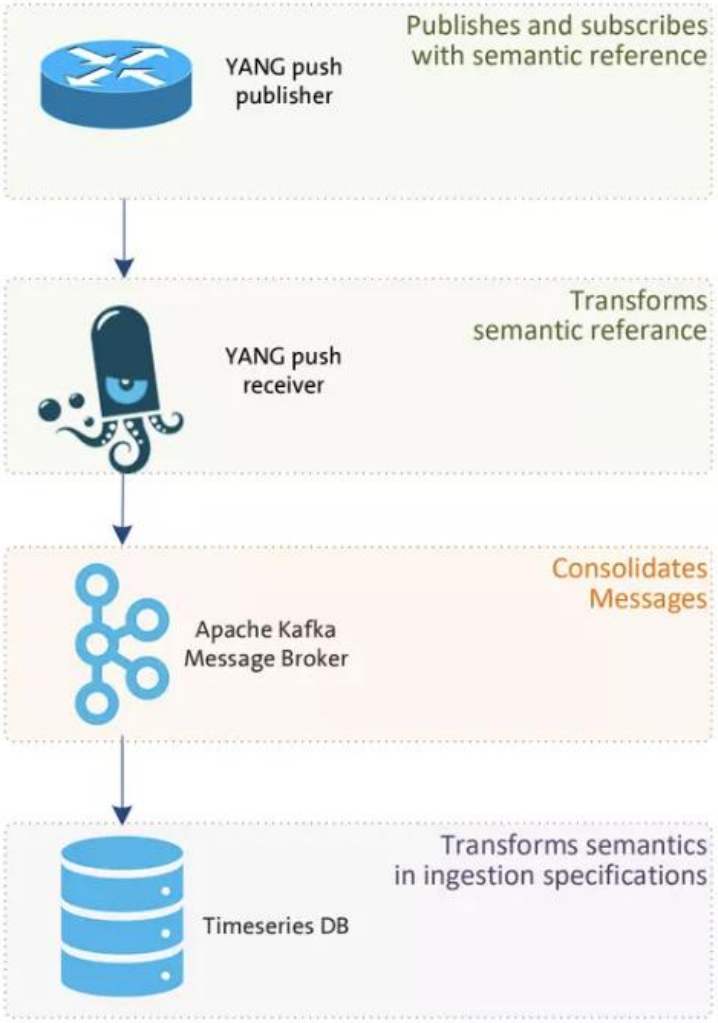
I. INTRODUCTION: Motivation

Network Analytics

Nowadays Network Analytics requires to correlate metrics from various sources to make **accurate detection/prediction**.

There are different sources and protocols to collect metrics. Here we focus on the **YANG push protocol**.

The metrics are the foundation to network analytics. We must have **reliable and meaningful data collection.**



Missing Semantics

In the TSDB, YANG model semantics might be lost. It is not clear how to interpret the data.

An example: Temperature

Data Mesh

The principle of Data Mesh is to present the data as a product. The quality of the data is guarantee by the team preparing the data.

In our case, the collected data comes with the YANG model.

I. INTRODUCTION: Overview

YANG Schema Registry

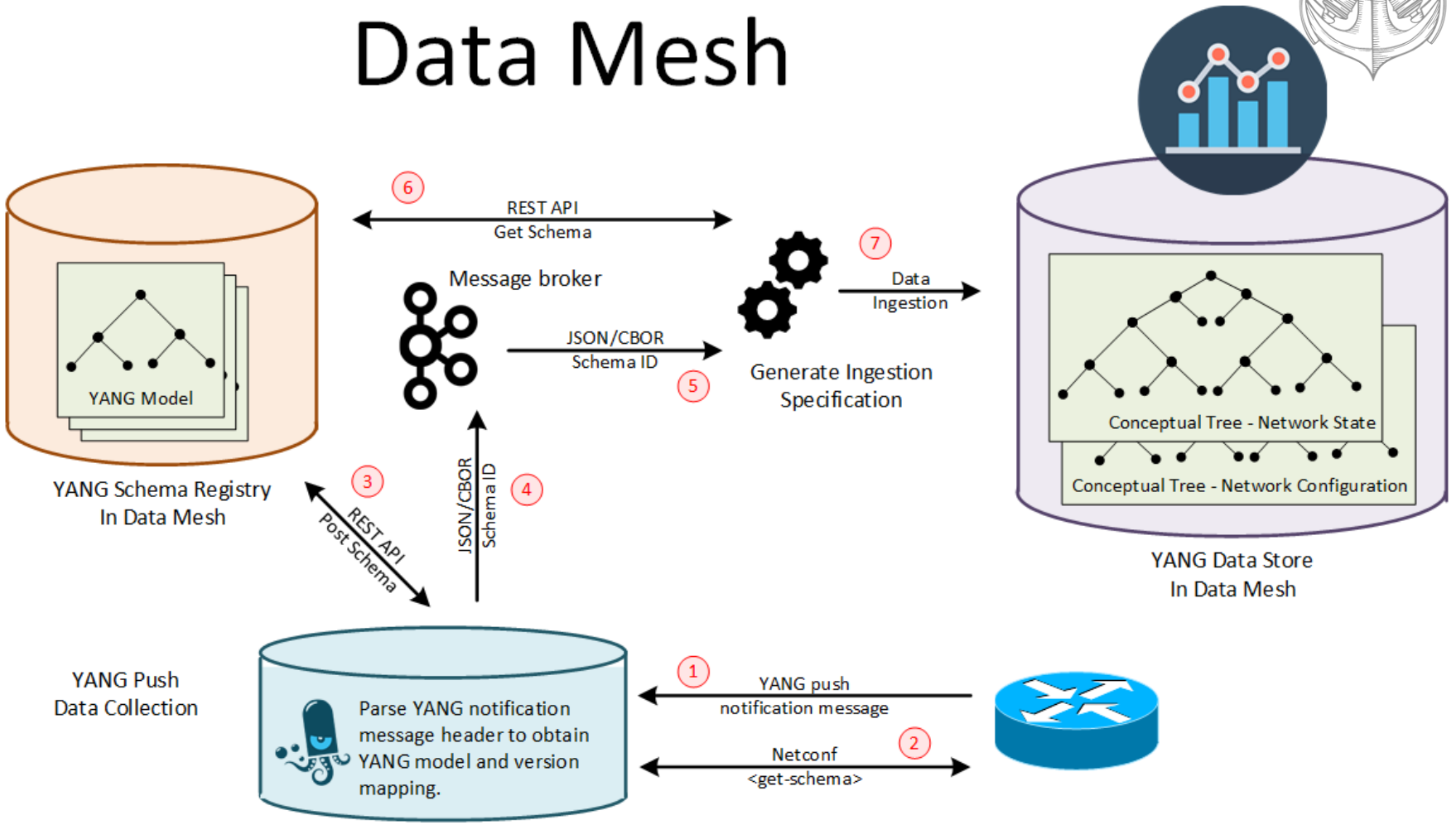
A solution to keep the semantics is to have a schema registry. Each message in Kafka contains a schema id pointing to the original YANG model.

Schema will be used during serialization and deseralization.

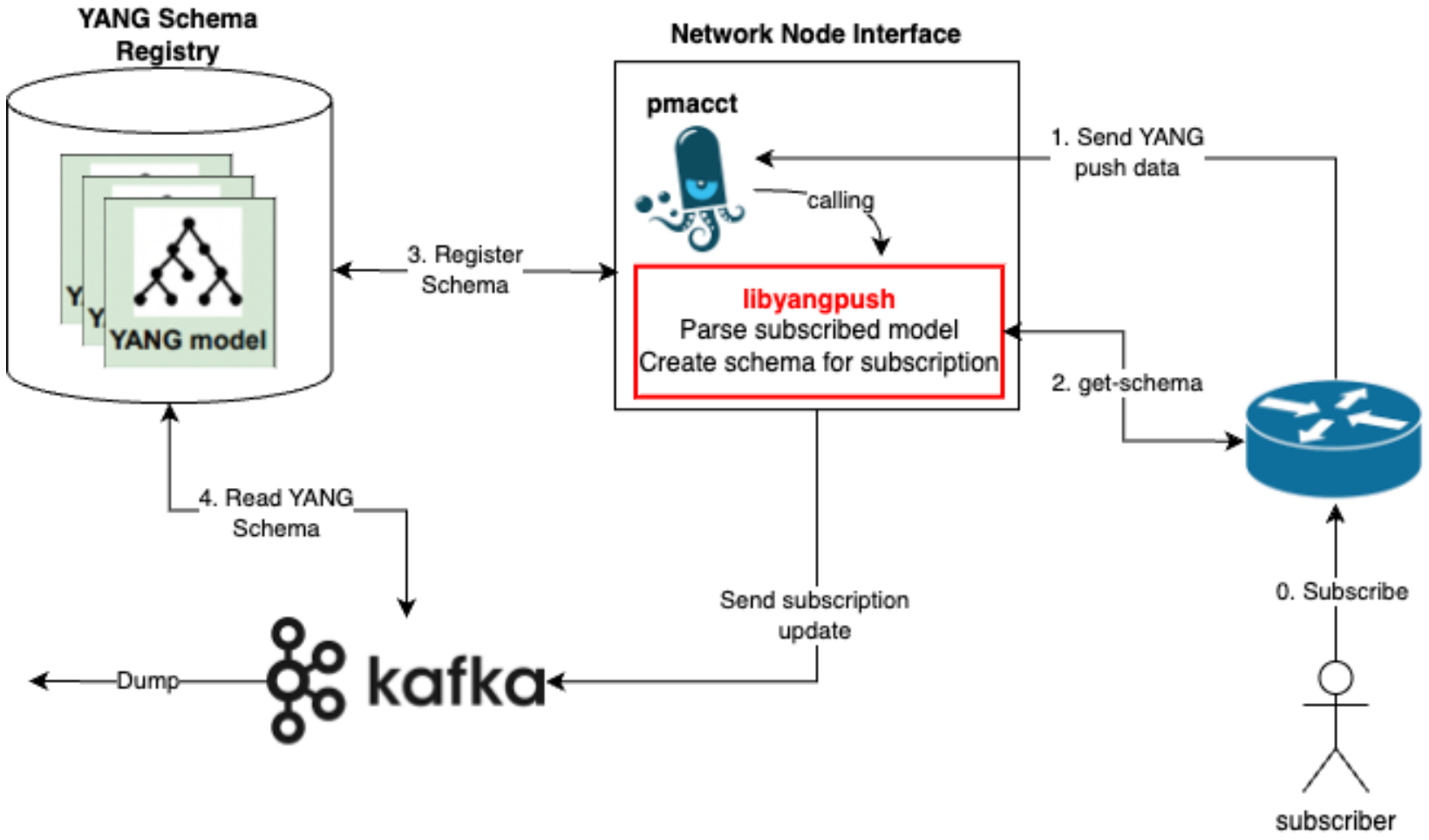
YANG push Receiver

Apart from collecting YANG data, the receiver needs to be extended to register schema for YANG subscription.

This is the goal of **libyangpush**.



I. INTRODUCTION: Overview and Current State



pmacct: A multi-purpose Network Monitoring tool(https://github.com/pmacct/pmacct)

Schema registry: Confluent pluggable schema registry has been extended to natively support YANG.

YANG push:

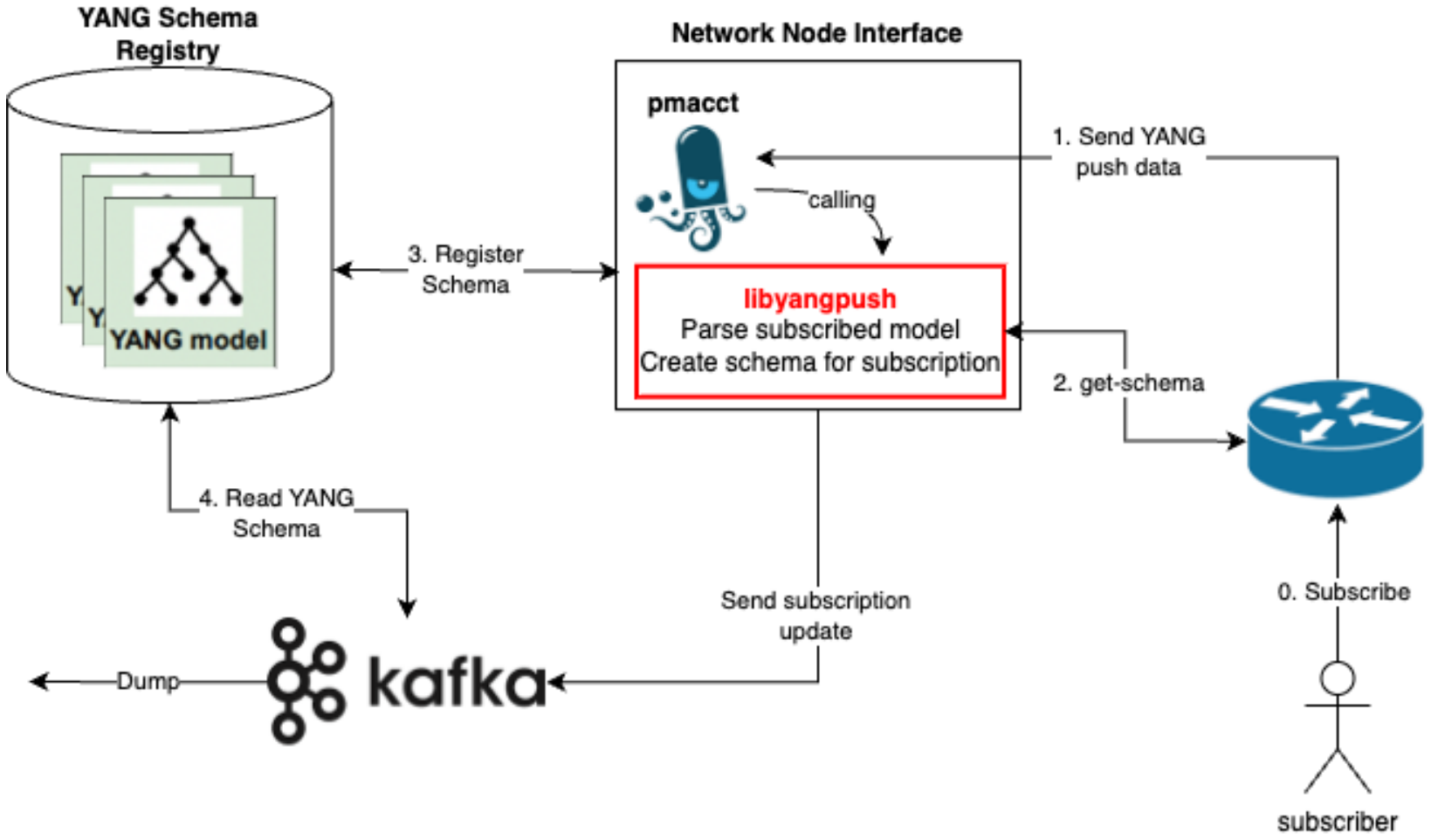
1. Subscription to YANG Notifications for Datastore Updates - RFC 8641

2. draft-ietf-netconf-udp-notif-11

It is partially being supported in the device

TABLE OF CONTENT

II. PROBLEMS



Problem 1:

How to store YANG model in schema registry?

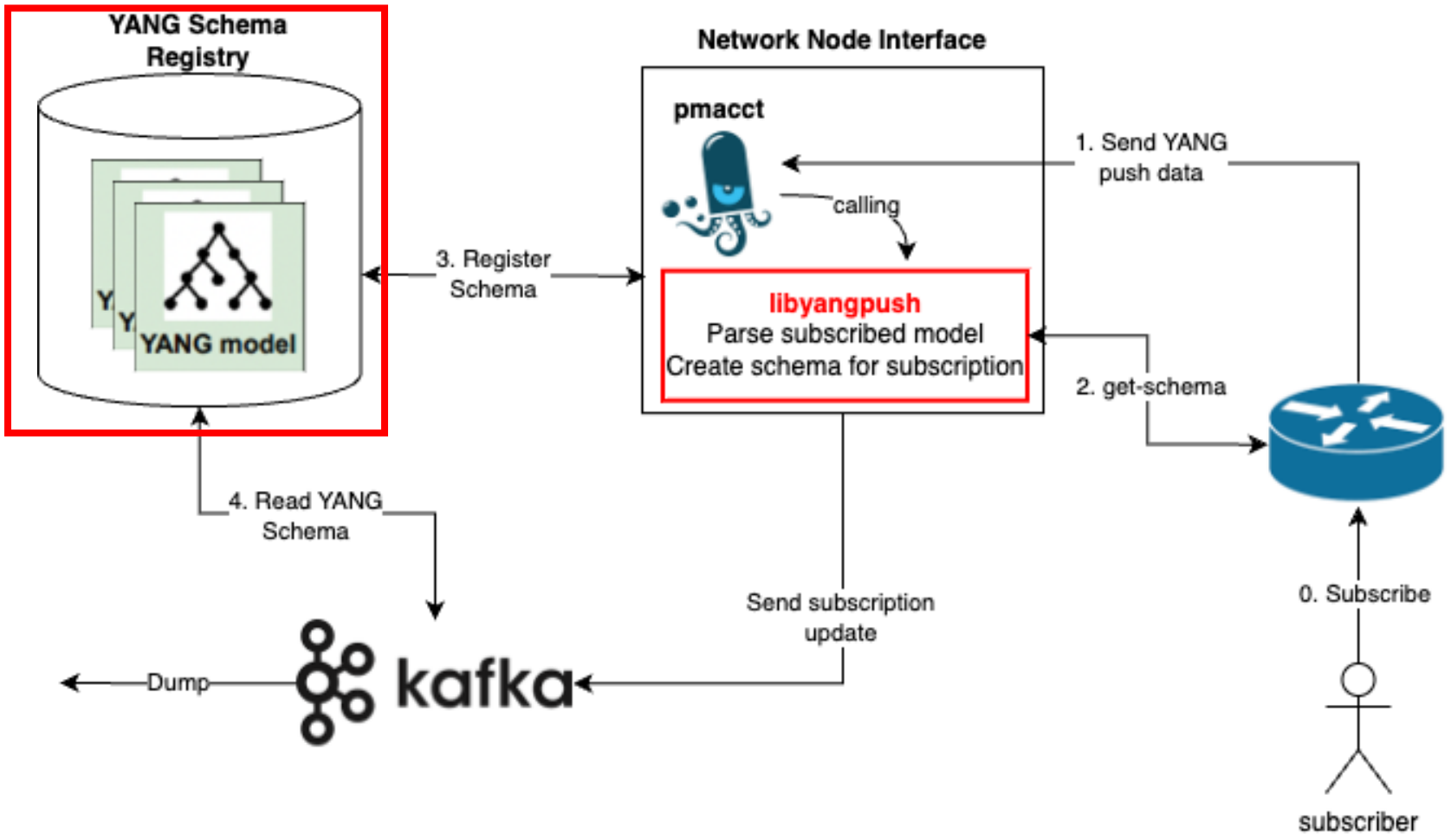
Problem 2:

How to know the subscribed model? (On the basis that the YANG push subscription is providing telemetry data)

Problem 3:

How to obtain the YANG model and its dependencies?

II. PROBLEM 1: How to store YANG model in schema registry?



Simulate YANG data structure:

Schema Content <-> YANG model code

Schema Reference <-> YANG model dependency

Functionality of schema registry:

Validate YANG model and its dependencies relationship

II. PROBLEM 2: How to know the subscribed model?

Solution 1:
Parse namespaces in the first YANG push message

It is possible to find the reverse dependency in this way.

```
"datastore-contents": {
    "ietf-interfaces:interfaces": [
     {
      "interface": {
       "name": "eth0",
       "type": "iana-if-type:ethernetCsmacd",
       "oper-status": "up",
       "speed": "1000000"
       "ietf-ip:ipv4": {
        "enabled": true,
        "forwarding": true
       }
      }
     }
    ]
}
```

Solution 2:
Use YANG to get subscription information:

Subscribed YANG models can be known by parsing fields: ***datastore-xpath-filter*** or ***datastore-subtree-filter***. The subscription is identified by a sub-id.

This information is exposed as YANG model. It can be obtain:

- **As a subscription.** Subscription will be synced through the first push-update, and updated through the push-change-update(new added, edit, remove etc.)
- **Via NETCONF <get> operation.** Send a <get> to obtain the same information.

```
<subscriptions>
  <subscription>
    <id>6666</id>
    <datastore>ds:operational</datastore>
    <datastore-xpath-filter>
       /a-module:a
    </datastore-xpath-filter>
    <encoding>encode-xml</encoding>
    <periodic>
      <period>30000</period>
    </periodic>
  </subscription>
</subscriptions>
```

II. PROBLEM 3: How to obtain the YANG model and its dependencies?

There are two possible implementation solutions, each with pros and cons

Solution 1:
On-demand Downloading

The idea of on-demand downloading is to send get-schema request to get the YANG models based on the model name we obtain. For the main model in the subscription, we can obtain its name using method in “identify model”.

Import and include can parsed from the YANG code. Deviate can be known by subscribing to ***/ietf-yang-library:modules-state*.**

Pros: schema is update-to-date

Cons: Cannot handle augment

```
module: ietf-yang-library
  +--ro yang-library
     |  +--ro module-set* [name]
     |     |  +--ro name                    string
     |     |  +--ro module* [name]
     |     |     |  +--ro name          yang:yang-identifier
     |     |     |  +--ro revision?     revision-identifier
     |     |     |  +--ro namespace     inet:uri
     |     |     |  +--ro location*     inet:uri
     |     |     |  +--ro submodule* [name]
     |     |     |     |  +--ro name          yang:yang-identifier
     |     |     |     |  +--ro revision?     revision-identifier
     |     |     |     |  +--ro location*     inet:uri
     |     |     |  +--ro feature*      yang:yang-identifier
     |     |     |  +--ro deviation*    -> ../../module/name
```

Solution 2:
Get-all-schema

The main idea of get-all-schemas is to get all models, store them in the disk, and analyse their full dependencies (import, include, deviate and augment) at the beginning of connection.

Pros: Can handle all dependencies

Cons: Downloading all schemas takes a lot of time

II. PROBLEM 3: How to obtain the YANG model and its dependencies?

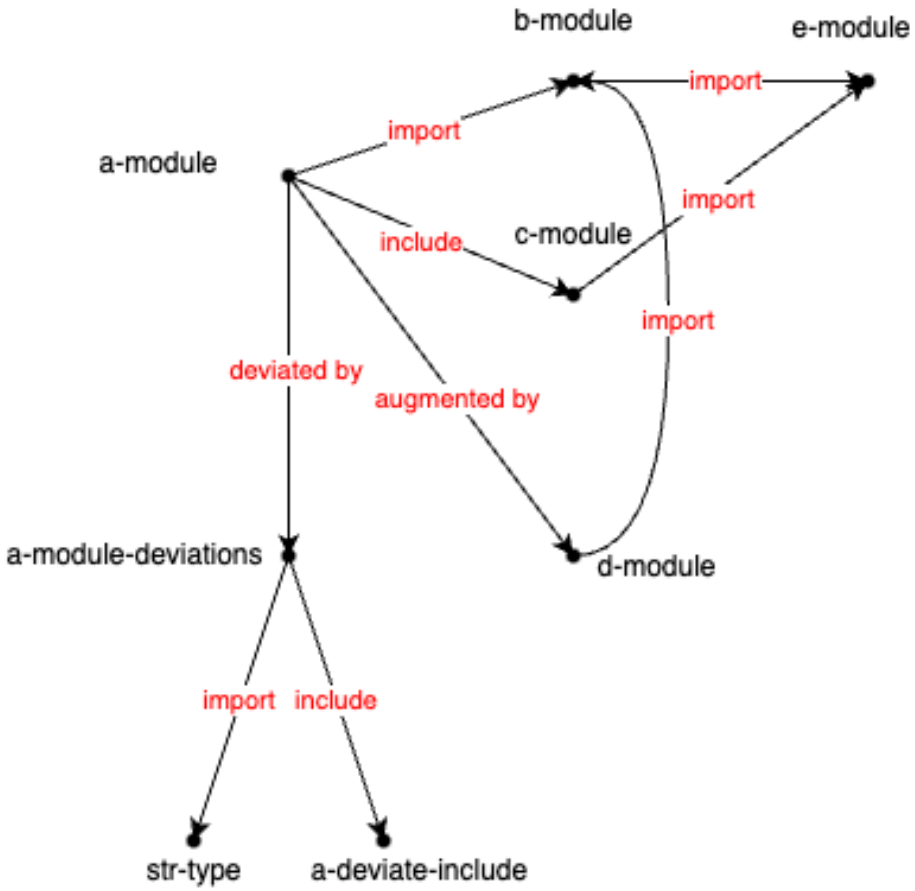
The YANG source can be obtained via NETCONF <get-schema>.
Now we need to obtain the full schema and dependencies.

Algorithm for finding dependency

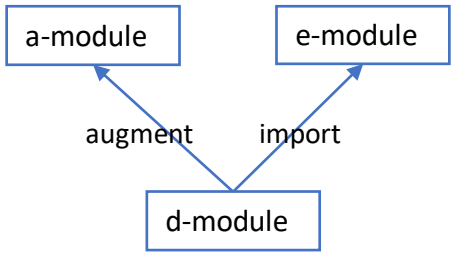
Expend on DFS to traverse the YANG model dependency tree, for the main model: sequentially search for its **import, include, augment and deviate**. For its dependency model, we only search for their **import and include**.

The model with the **greatest depth** will be put into register list first, then the top level module, in order to ensure that each register model has the dependency to refers to in the schema registry.

A hash map is used throughout the traverse with the index being djb2 hash of the model name to make sure that the model will not be put into register list twice.



II. PROBLEMS: An example for the chosen solution




```
<subscriptions>
  <subscription>
    <id>6666</id>
    <datastore>ds:operational</datastore>
    <datastore-xpath-filter>
       /a-module:a
    </datastore-xpath-filter>
    <encoding>encode-xml</encoding>
    <periodic>
      <period>30000</period>
    </periodic>
  </subscription>
</subscriptions>
```

```
module: a-module
 +--rw a
   +--rw a-instance* [name]
   |  +--rw name     string
   |  +--rw state?   string
   +--rw d:y
     +--rw d:y-leaf?   e:e-enum
```

The schema for a-module require to register a-modules, e-module and d-module.

Schema registration Order:

- a-module, reference{}
- e-module, reference{}
- d-module, reference{e-module, a-module}
- a-module, reference{d-module, e-module}

TABLE OF CONTENT

III. Proposal: Augmenting YANG model *ietf-yang-library*

Problem 3:

How to obtain the YANG model and its dependencies?

Explanation:

As for the current chosen solution, we cannot invalidate the model stored in disk when they have been updated in the device. As a way to get the most up-to-date reverse dependency model, the on-demand downloading only support checking for deviations currently. However, it is reasonable to augment to make the ietf-yang-library to also support storing the augmentations.

In this way, we can get rid of having to use the get-all-schemas solution, while we can also handle the reverse dependencies easily.

```
module: ietf-yang-library
  +--ro yang-library
  |  +--ro module-set* [name]
  |  |  +--ro name                    string
  |  |  +--ro module* [name]
  |  |  |  +--ro name          yang:yang-identifier
  |  |  |  +--ro revision?     revision-identifier
  |  |  |  +--ro namespace     inet:uri
  |  |  |  +--ro location*     inet:uri
  |  |  |  +--ro submodule* [name]
  |  |  |  |  +--ro name          yang:yang-identifier
  |  |  |  |  +--ro revision?     revision-identifier
  |  |  |  |  +--ro location*     inet:uri
  |  |  |  +--ro feature*      yang:yang-identifier
  |  |  |  +--ro deviation*    -> ../../module/name
```

```
module: ietf-yang-library-add-augmentation-01
  +--ro yang-library
  |  +--ro module-set* [name]
  |  |  +--ro name                    string
  |  |  +--ro module* [name]
  |  |  |  +--ro name            yang:yang-identifier
  |  |  |  +--ro revision?       revision-identifier
  |  |  |  +--ro namespace       inet:uri
  |  |  |  +--ro location*       inet:uri
  |  |  |  +--ro submodule* [name]
  |  |  |  |  +--ro name          yang:yang-identifier
  |  |  |  |  +--ro revision?     revision-identifier
  |  |  |  |  +--ro location*     inet:uri
  |  |  |  +--ro feature*         yang:yang-identifier
  |  |  |  +--ro deviation*       -> ../../module/name
  |  |  |  +--ro augmentation*    -> ../../module/name
```

IV. Conclusion & Future Work

Outcomes

Demo is presented during **IETF117 Hackathon**: https://wiki.ietf.org/en/meeting/117/hackathon#network-telemetry-yang-push-integration-into-apache-kafka

Demonstrate the interaction with a working YANG schema registry during the demo

Github repository for **libyangpush:**

[github link will be put here]

Gaps to fill

- Integration with pmacct
- Test with device as YANG push becomes better supported
- Augment YANG model ***ietf-yang-library*** to support the record of augmentations(to support the on-demand downloading)

Future Work

- Deploy the functionalities into Swisscom lab.

Industry Post:

https://github.com/graf3net/draft-daisy-kafka-yang-integration/blob/main/draft-daisy-kafka-yang-integration-05.md

https://www.linkedin.com/pulse/network-analytics-ietf-115-london-thomas-graf/

https://www.linkedin.com/pulse/network-analytics-ietf-116-yokohama-thomas-graf/

https://www.linkedin.com/pulse/network-analytics-ietf-117-san-francisco-thomas-graf/