CBOR Encoding for HTTPS-based Transport for YANG Notifications

https://datatracker.ietf.org/doc/html/draft-chittapragada-netconf-https-notif-cbor

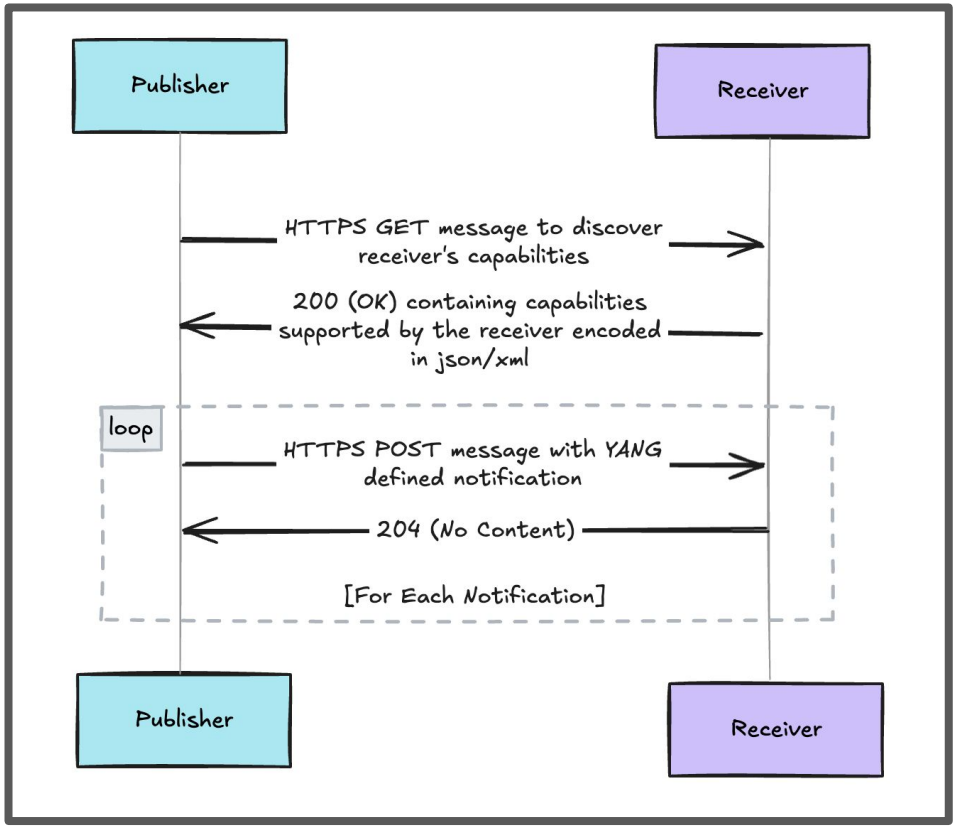
Bharadwaja Meherrushi C, Hayyan Arshad, Siddharth Bhat, Vartika T Rao, Mohit P. Tahiliani

National Institute of Technology Karnataka, Surathkal, India

IETF 123, Madrid

Overview of the https-notif-draft

- Publisher sends a `GET` request to `/capabilities` resource of the Receiver (i.e., Collector)
- Publisher learns the Receiver's capabilities from the response body.
- Publisher sends a `POST` request with YANG defined notification data.
- Receiver responds with a `204 (No Content)` is successful.
- JSON and XML support for encoding



Why CBOR Extension?

Using HTTPS, which is a secure form of HTTP Semantics [RFC9110], **maximizes transport-level interoperability**, while allowing for a **variety of encoding options**. Current draft supports JSON and XML encoding as options .

CBOR is another **widely used encoding standard** and it also provides **greater throughput in low bandwidth networks.**

Our draft describes the **standard to conform to in order to include** the **CBOR** encoding format.

In accordance with Encoding of Data Modeled with YANG in the Concise Binary Object Representation (CBOR) [RFC9254], we propose support for usage of both **SIDs in keys** and **Names in keys**.

Overview of Draft

```
Diagnostic notation:
{
   "ietf-https-notif:notification":{

"eventTime":"2013-12-21T00:01:00Z",
       "example-mod:event":{
       "event-class":"fault",
       "reporting-entity":{
              "card":"Ethernet0"
       },
       "severity":"major"
       }
   }
}
```

Examples of the `GET` and `POST` request and reply encoded in CBOR along with their diagnostic notation and encoding examples

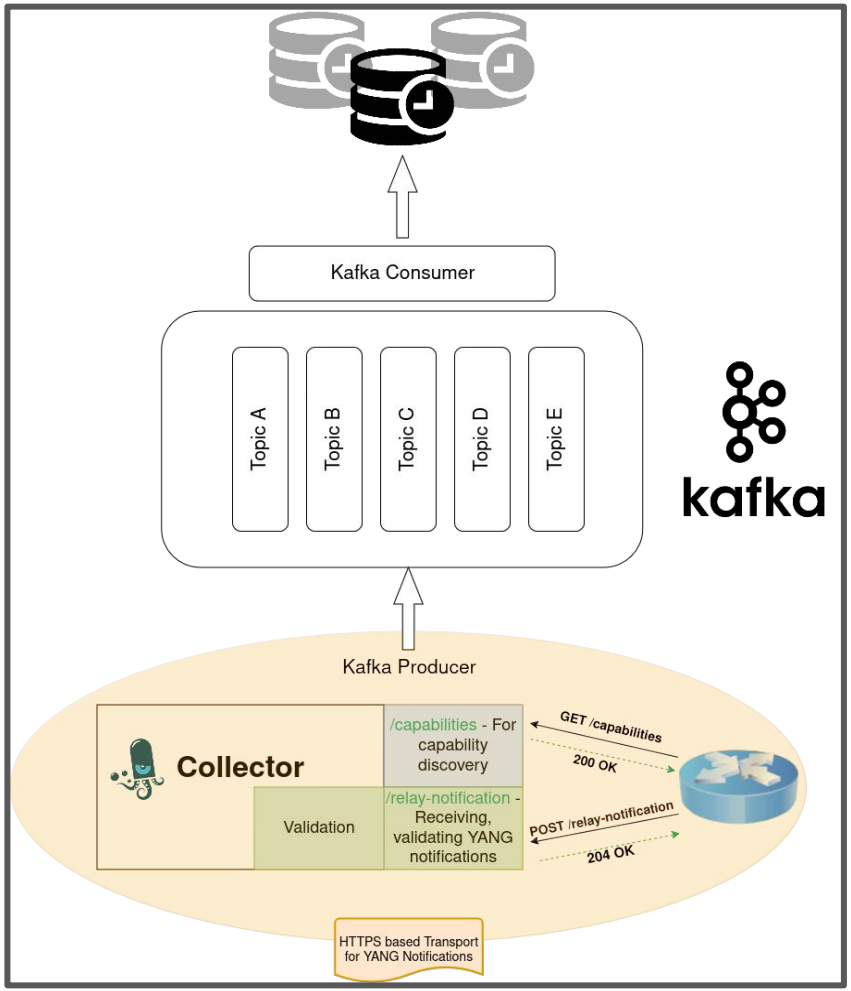
YANG notifications can be encoded in **CBOR** using **Names or SIDs in keys.**

```
Cbor Encoding:

A1                                      # map(1)
   78 1D                                # text(29)

696574662D68747470732D6E6F7469663A6E6F74696669636174696F
6E # "ietf-https-notif:notification"
   A2                                   # map(2)
      69                                # text(9)
         6576656E7454696D65             # "eventTime"
      74                                # text(20)
         323031332D31322D32315430303A30313A30305A #
"2013-12-21T00:01:00Z"
      71                                # text(17)
         6578616D706C652D6D6F643A6576656E74 #
"example-mod:event"
      A3                                # map(3)
         68                             # text(8)
            7365766572697479            # "severity"
         65                             # text(5)
            6D616A6F72                  # "major"
         6B                             # text(11)
            6576656E742D636C617373      # "event-class"
         65                             # text(5)
            6661756C74                  # "fault"
         70                             # text(16)
            7265706F7274696E672D656E74697479 #
"reporting-entity"
         A1                             # map(1)
            64                          # text(4)
               63617264                 # "card"
            69                          # text(9)
               45746865726E657430       # "Ethernet0"
```

Implementation Details

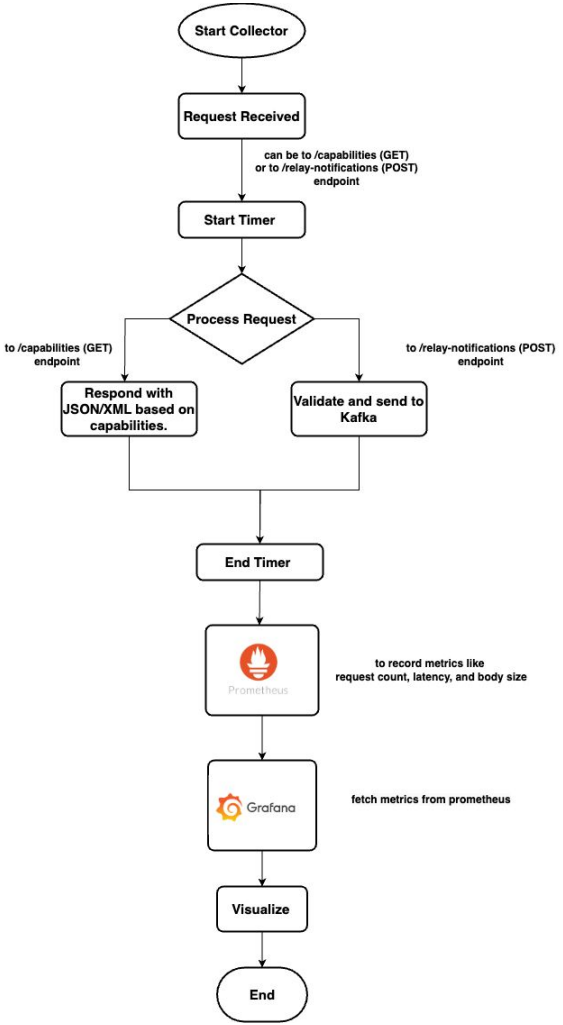
Implementation Architecture

- Collector (receiver) supports `/capabilities` and `/relay-notifications`
- Collector receives the relay notification and **publishes the notification data to a designated Kafka topic**.
- A parallel Kafka **consumer continuously reads messages from this topic and inserts the data into a Time Series Database** (InfluxDB in our implementation)
- **The entire code is containerized!**



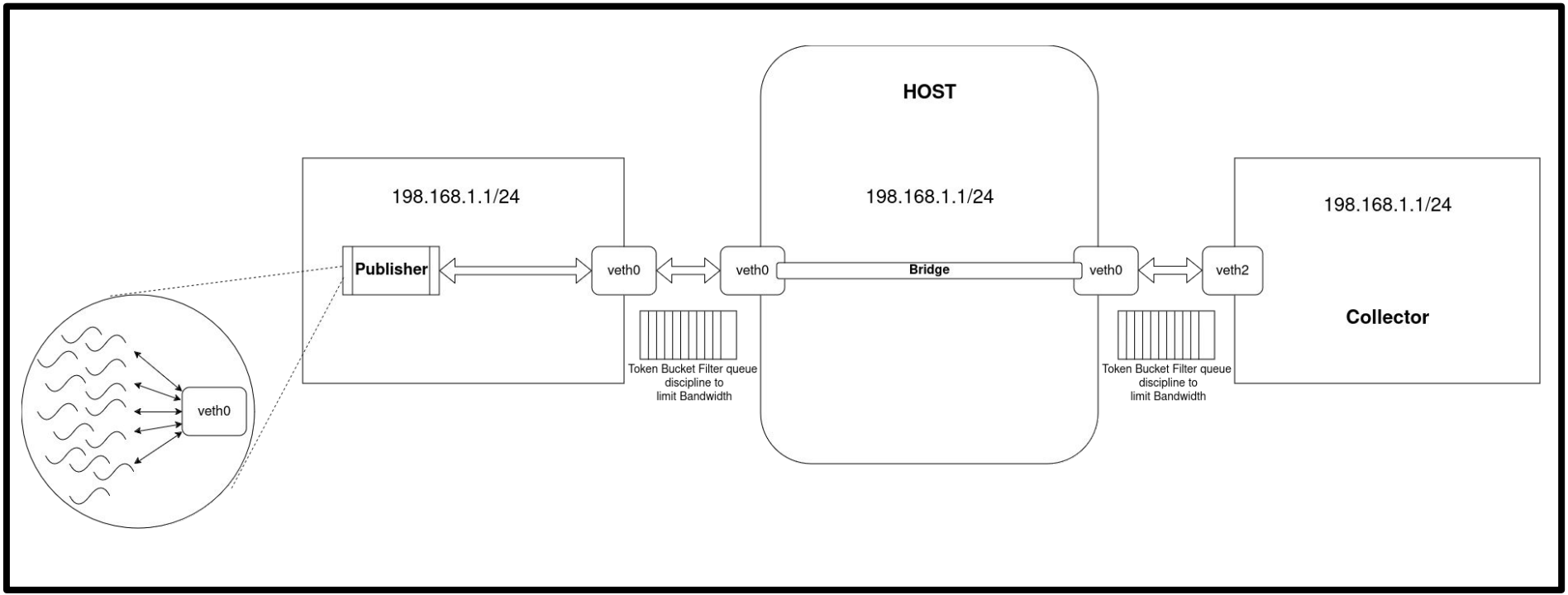
Monitoring metrics with Prometheus and Grafana

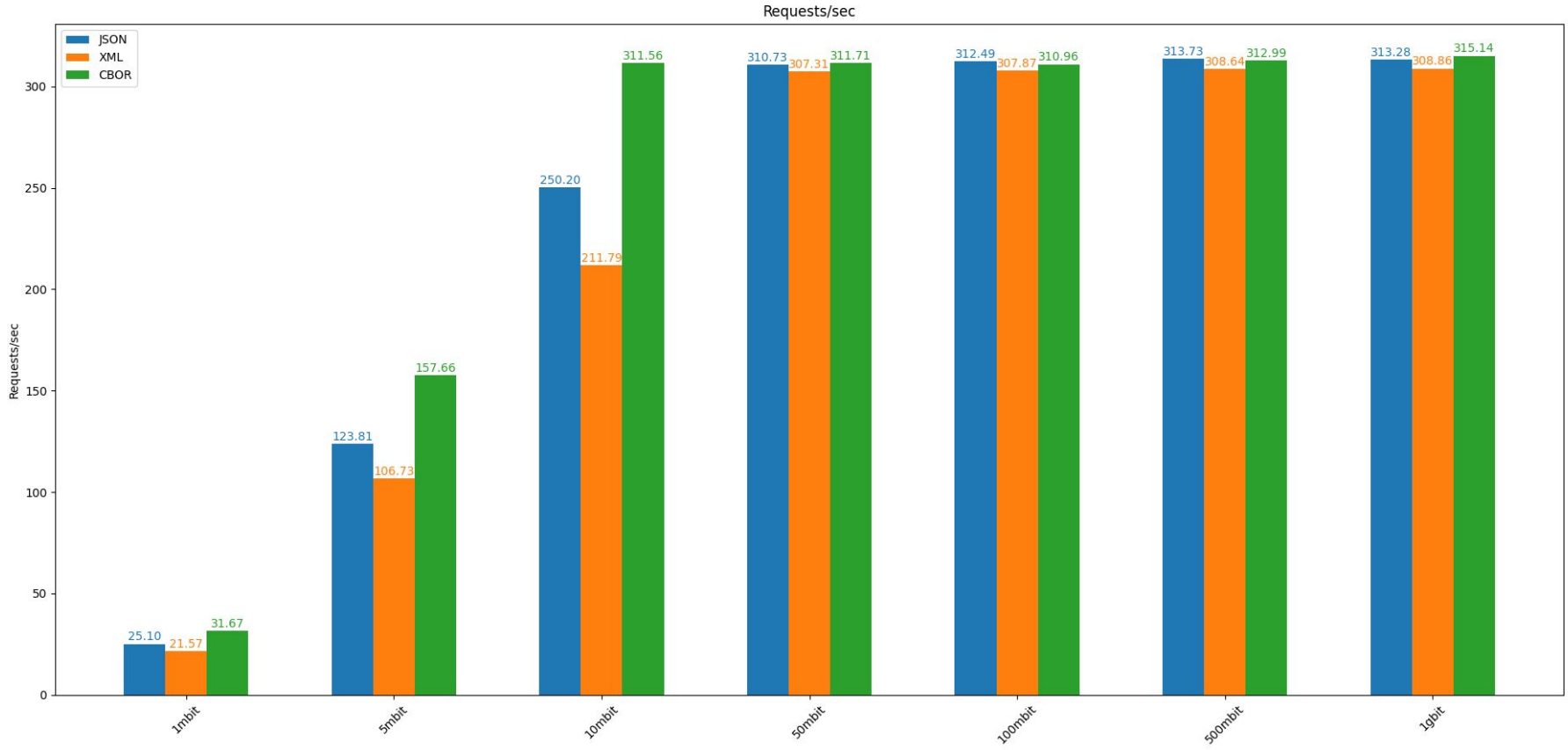
- The application uses Prometheus client libraries to define **metrics such as HTTPS request counts, request latencies and POST request body sizes**.
- Prometheus collects the metrics exposed by the application. Grafana is used to visualize these metrics.



Performance Analysis of Encoding Formats Under Varying Bandwidth

Virtual Network Namespace Setup





Req/sec, average measurement of the above experiment

Challenges Faced

- Lack of tooling and support for YANG in general.
  E.g.: No support for YANG Data Structure Extensions Module (sx: *structure*) in yangson (Python)
- Lack of support for XML and CBOR YANG data validation in libraries and tooling like yangson and libyang [we had to convert them to JSON! which added unnecessary overhead and affected our numbers :( ]
- We have worked towards adding CBOR support in Libyang during the hackathon(https://datatracker.ietf.org/meeting/123/materials/slides-123-hackathon-sessd-adding-cbor-support-in-libyang-00)
- No YANG data model defined (*yet!)* for notification data. This also means we are unable to request for SIDs from IANA for the same.

Any Questions?

Thank you!

Implementation: https://github.com/MeherRushi/https-notif-draft-impl/

Internet Draft:
https://datatracker.ietf.org/doc/draft-chittapragada-netconf-https-notif-cbor/

Pls give it a read and any feedback is welcome :)

A special thanks to Mahesh J and Kent Watsen, the authors of the HTTPS-notif-draft for their guidance and support