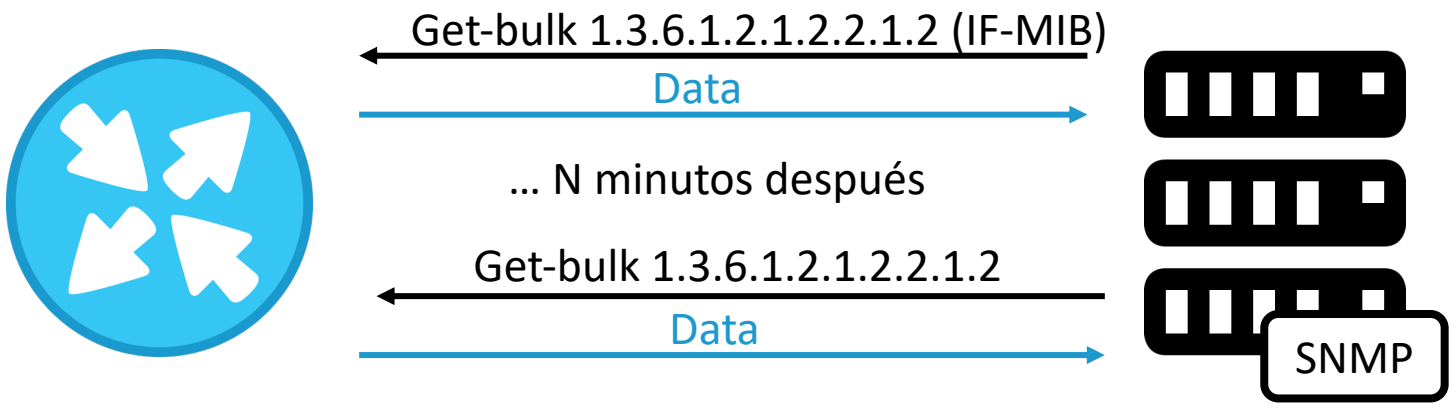
El lado oscuro del Streaming Telemetry

Camilo Cardona

NTT

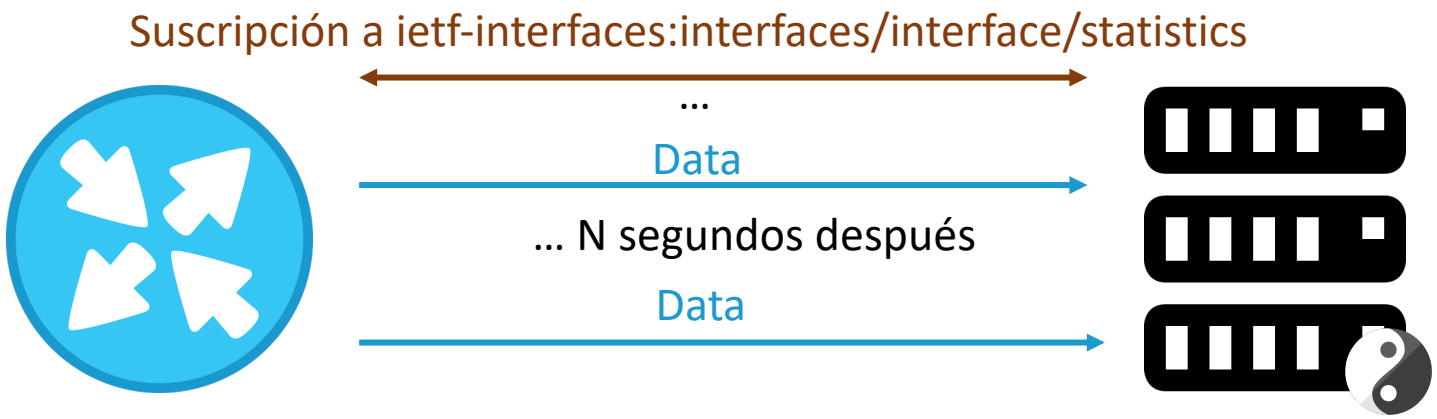
¿Qué es el streaming telemetry?

- Streaming de valores definidos en un modelo **YANG**
- Alternativa al SNMP, y, sobretodo, al polling
- Varias alternativas en las tecnologías que lo soportan



¿Qué es el streaming telemetry?

- Streaming de valores definidos en un modelo **YANG**
- Alternativa al SNMP, y, sobretodo, al polling
- Varias alternativas en las tecnologías que lo soportan



¿Cuál es la desventaja de del SNMP?

- No habrá nuevo Contenido: Las métricas de nuevas tecnologías no se expondrán en MIBs (i.e. SMIv2) sino en módulos de YANG.
- Escalabilidad: Recolectar mas datos de manera precisa
  - Referencia: “SNMP is dead” de NANOG 73*

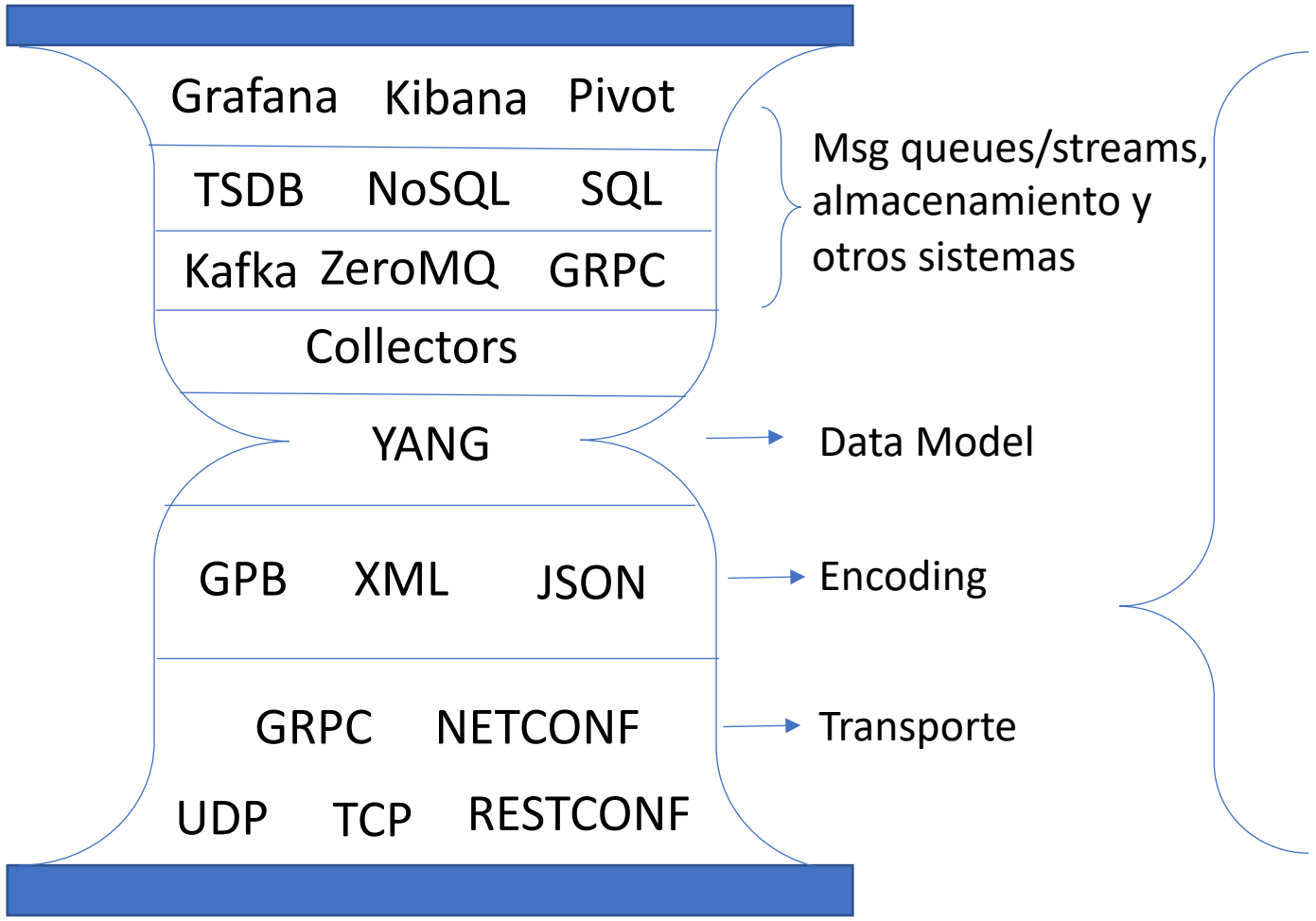
¿Esto es suficiente para convencer a la industria para abandonar el SNMP?

*Referencias en la ultima slide

Estado del streaming telemetry en la industria

- Se ha desarrollado en los últimos años, pero está fracturado
- Openconfig / IETF / propietario
  - Openconfig se ha desarrollado rápidamente, el código disponible es mas amplio, pero es básicamente controlado por Google
  - La IETF ha trabajado lentamente. Definió un framework flexible, pero hay poco código

Ecosistema del streaming telemetry



- Plano de control
  - Cómo se configuran/establecen los flujos de datos (streams)
- Tipos de flujos de datos:
  - Sample
  - On-change

Transporte

- Define el protocolo de comunicación entre el dispositivo y el colector
- Ejemplos:
  - Protocolos propietarios basados en UDP, TCP
  - GRPC
    - Definiciones propietarias
    - GNMI
  - NETCONF (RFC 8640), RESTCONF

Transporte

- Ejemplos: UDP, TCP, GRPC (e.g. GNMI), NETCONF (RFC 8640), RESTCONF
- En algunos de estos la comunicación la establece el servidor, en otras el colector. Algunos permiten ambas.
- La capa de transporte define el control de flujo (i.e. backpressure). **Los protocolos complejos no son puramente PUSH.**
- GNMI define como detectar y tratar los consumidores lentos
  - Mirar “duplicates” en la sección 2.1 de la gnmi-specification*

Encoding

- Ejemplos: XML, JSON, Google Protocol Buffers (GPB), otros pueden venir…
- GPB puede ser genérico (e.g. GPB-KV) o pueden existir esquemas para cada modelo (e.g. GPB Compact)
- Todo es un trade-off:
  - Recomendado: Capitulo 4. del libro Designing Data-Intensive Applications
- Ya es relativamente sencillo decodificar/codificar en varios lenguajes
- El tipo de encoding es importante cuando se desea escalar.

Muchos enrutadores solo soportan un tipo de encoding. Los colectores deben ser flexibles.

YANG modelos

- Modelos standard
  - Standards, pero no siempre cumplen con todas las necesidades
- Modelos nativos
  - Exclusivos de un vendor/modelo
  - Pueden cubrir muchos mas datos
  - Muchas veces auto-generados y con problemas en estructura (e.g. Excesivamente jerárquicos, claves no bien definidas)

Los colectores posiblemente deban transformar los datos para ajustar modelos

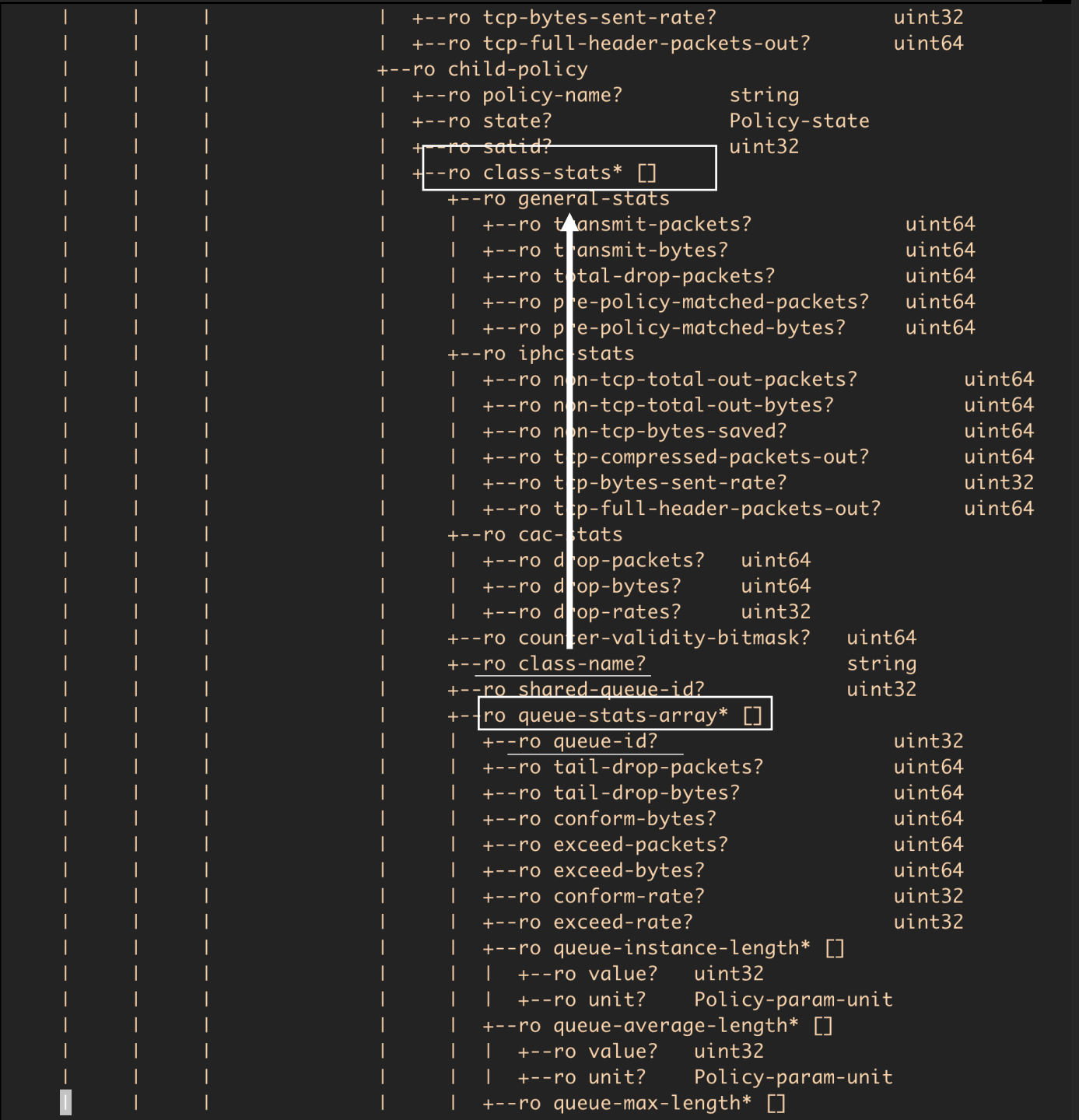
```
module: openconfig-interfaces
  +--rw interfaces
     +--rw interface* [name]
        +--rw name              -> ../config/name
        +--rw config
        |  +--rw type           identityref
        |  +--rw mtu?           uint16
        |  +--rw name?          string
        |  +--rw description?   string
        |  +--rw enabled?       boolean
              Edited
        +--rw subinterfaces
           +--rw subinterface* [index]
              +--rw index     -> ../config/index
              +--rw config
              |  +--rw index?         uint32
              |  +--rw name?          string
              |  +--rw description?   string
              |  +--rw enabled?       boolean
              +--ro state
                 +--ro index?          uint32
                 +--ro name?           string
                 +--ro description?    string
                 +--ro enabled?        boolean
                 +--ro ifindex?        uint32
                 +--ro admin-status    enumeration
                 +--ro oper-status     enumeration
                 +--ro last-change?    yang:timeticks
                 +--ro counters
                    +--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:counter64
                    +--ro in-errors?           yang:counter64
                    +--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:counter64
                    +--ro out-errors?          yang:counter64
                    +--ro last-clear?          yang:date-and-time
```

Data


```
{
  "collector":{
    "data":{
      "collection_end_time":1568118857420,
      "collection_id":"5778",
      "collection_start_time":1568118857420,
      "collection_timestamp":1568119008859,
      "encoding_path":"openconfig-interfaces:interfaces",
      "encoding_type":"None",
      "msg_timestamp":1568118857502,
    },
    "grpc":{
      "grpcPeer":"",
      "ne_vendor":""
    }
  },
  "interfaces":{
    "interface":[
      {
        "name":"parent-interface",
        "subinterfaces":{
          "subinterface":[
            {
              "index":"1",
              "state":{
                "admin_status":1,
                "description":"",
                "enabled":true,
                "ifindex":42,
                "index":1,
                "last_change":"1566458191854000000",
                "logical":false,
                "name":"subinterface1",
                "oper_status":0
              }
            }
          ]
        }
      }
    ]
  }
}
```

```
module: Cisco-IOS-XR-qos-ma-oper
  +--ro qos
     +--ro nv-interface-table
     |  +--ro interface* [interface-name]
     |     +--ro nodes
     |     |  +--ro node* [node-name]
     |     |     +--ro node-name    xr:Node-id
     |     |     +--ro input
     |     |     |  +--ro service-policy-names
     |     |     |     +--ro service-policy-instance* [service-policy-name]
     |     |     |        +--ro service-policy-name    xr:Cisco-ios-xr-string
     |     |     |        +--ro statistics
     |     |     |           +--ro policy-name?         string
     |     |     |           +--ro state?               Policy-state
     |     |     |           +--ro satid?               uint32
     |     |     |           +--ro class-stats* []
     |     |     |              +--ro general-stats
     |     |     |              |  +--ro transmit-packets?            uint64
     |     |     |              |  +--ro transmit-bytes?              uint64
     |     |     |              |  +--ro total-drop-packets?          uint64
     |     |     |              |  +--ro pre-policy-matched-bytes?    uint64
     |     |     |              +--ro iphc-stats
     |     |     |              |  +--ro non-tcp-total-out-packets?        uint64
     |     |     |              |  +--ro non-tcp-bytes-sent-rate?          uint32
     |     |     |              |  +--ro tcp-bytes-sent-rate?              uint32
     |     |     |              |  +--ro tcp-full-header-packets-out?      uint64
     |     |     |              +--ro child-policy
     |     |     |              |  +--ro policy-name?         string
     |     |     |              |  +--ro state?               Policy-state
     |     |     |              |  +--ro satid?               uint32
     |     |     |              |  +--ro class-stats* []
     |     |     |              |     +--ro general-stats
     |     |     |              |     |  +--ro transmit-packets?            uint64
     |     |     |              |     |  +--ro transmit-bytes?              uint64
     |     |     |              |     |  +--ro total-drop-packets?          uint64
     |     |     |              |     |  +--ro pre-policy-matched-packets?  uint64
     |     |     |              |     |  +--ro pre-policy-matched-bytes?    uint64
     |     |     |              |     +--ro iphc-stats
     |     |     |              |     |  +--ro non-tcp-total-out-packets?        uint64
     |     |     |              |     |  +--ro non-tcp-total-out-bytes?          uint64
     |     |     |              |     |  +--ro non-tcp-bytes-saved?              uint64
     |     |     |              |     |  +--ro tcp-compressed-packets-out?       uint64
     |     |     |              |     |  +--ro tcp-bytes-sent-rate?              uint32
```

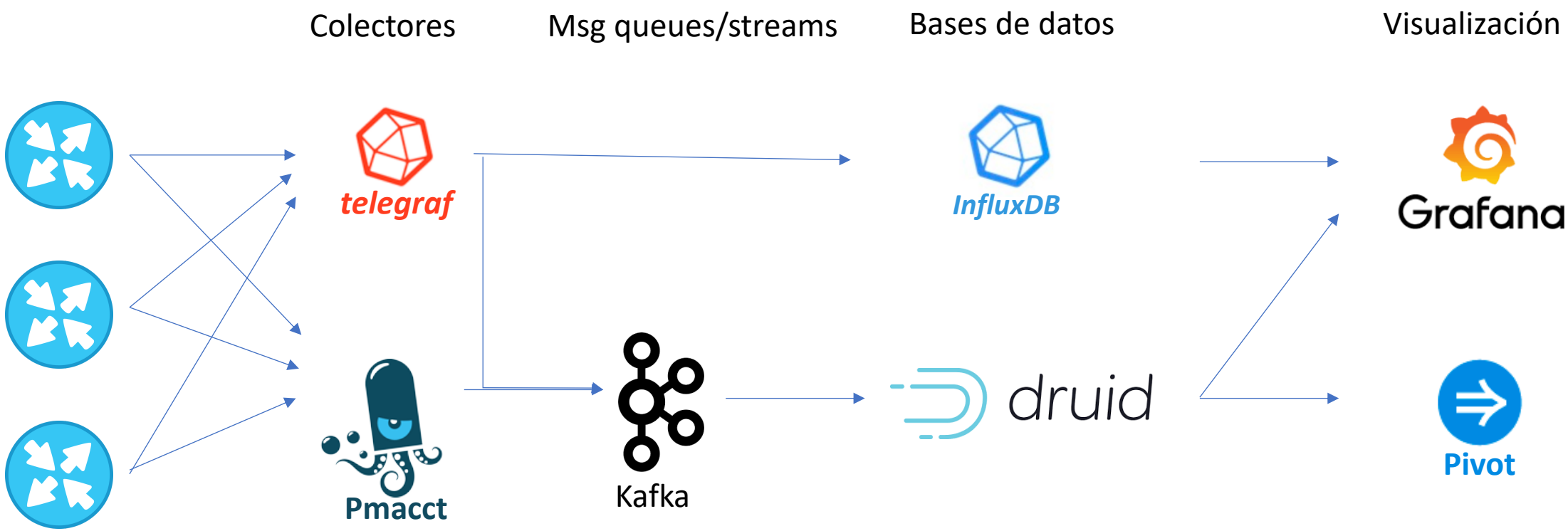
Y el key?


+--ro tcp-bytes-sent-rate? uint32
+--ro tcp-full-header-packets-out? uint64
+--ro child-policy
+--ro policy-name? string
+--ro state? Policy-state
+--ro satid? uint32
+--ro class-stats* []
+--ro general-stats
+--ro transmit-packets? uint64
+--ro transmit-bytes? uint64
+--ro total-drop-packets? uint64
+--ro pre-policy-matched-packets? uint64
+--ro pre-policy-matched-bytes? uint64
+--ro iphc-stats
+--ro non-tcp-total-out-packets? uint64
+--ro non-tcp-total-out-bytes? uint64
+--ro non-tcp-bytes-saved? uint64
+--ro tcp-compressed-packets-out? uint64
+--ro tcp-bytes-sent-rate? uint32
+--ro tcp-full-header-packets-out? uint64
+--ro cac-stats
+--ro drop-packets? uint64
+--ro drop-bytes? uint64
+--ro drop-rates? uint32
+--ro counter-validity-bitmask? uint64
+--ro class-name? string
+--ro shared-queue-id? uint32
+--ro queue-stats-array* []
+--ro queue-id? uint32
+--ro tail-drop-packets? uint64
+--ro tail-drop-bytes? uint64
+--ro conform-bytes? uint64
+--ro exceed-packets? uint64
+--ro exceed-bytes? uint64
+--ro conform-rate? uint32
+--ro exceed-rate? uint32
+--ro queue-instance-length* []
+--ro value? uint32
+--ro unit? Policy-param-unit
+--ro queue-average-length* []
+--ro value? uint32
+--ro unit? Policy-param-unit
+--ro queue-max-length* []

¿Qué hacer con los datos que se generan?

- Surgirán herramientas para requerimientos ”standard”
  - Contadores de interfaces
  - Modelos/transporte/Encoding standard
- Cada ISP/Red tendrá que ajustar sus herramientas para requerimientos ”ad-hoc”

¿Que hacer con los datos que se generan?



Conclusiones

- La industria todavía está en maduración
  - Sin embargo, ya es posible recolectar datos útiles de modelos básicos
- Los colectores deben ser flexibles.
- Todavía se necesita trabajo en la estandarización y afinación de modelos
  - Post-transformaciones pueden ser necesarias
- No es trivial, son necesarias mejores herramientas.
- Probablemente el mejor recurso para aprender todo esto en un solo lugar:
  - Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI.

Referencias

- Icons:
  - Router: **Yudha Agung Pribadi**, https://www.iconfinder.com/icons/575175/cisco_networking_router_stencil_visio_icon
  - Server: https://www.iconfinder.com/icons/298866/server_icon
  - Yang: https://www.iconfinder.com/icons/379381/yang_yin_icon
- Presentaciones:
  - https://pc.nanog.org/static/published/meetings/NANOG73/1677/20180625_Shakir_Snmp_Is_Dead_v1.pdf
- Otros
  - Especificación GNMI: https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md

Back-up

Tipos de Stream

- Sampling
  - Los datos son enviados a una frecuencia determinada
- On-change
  - Los datos se envían solamente cuando hay cambios
  - Eficiente para unos tipos de datos, pero complica el sistema

Control plane

- YANG PUSH (en RFC8639) define una manera estándar de establecer subscripciones
  - Independiente de transporte
- GNMI define sus propios mecanismos de suscripción.