ReTEC

Revista Electrónica de
Tecnología, Educación y Ciencia
ISSN: 2953-5654
http://retec.unsa.edu.ar
Universidad Nacional de Salta

Hacia un Nuevo Paradigma en la Telemetría de Redes

Ernesto Sánchez, Daniel Arias Figueroa

Facultad de Ingeniería – Universidad Católica de Salta
C.I.D.I.A. – Centro de Investigación y Desarrollo en Informática Aplicada
Facultad de Ciencias Exactas – Universidad Nacional de Salta
{ esanchez , daaf }@cidia.unsa.edu.ar

Revista Electrónica de Tecnología, Educación y Ciencia,

Volumen 1, Número 3, pág. 71-90, jun, 2026. ISSN: 2953-5654

Disponible en http://retec.unsa.edu.ar/

Hacia un Nuevo Paradigma en la Telemetría de Redes

Ernesto Sánchez, Daniel Arias Figueroa

Facultad de Ingeniería – Universidad Católica de Salta
C.I.D.I.A. – Centro de Investigación y Desarrollo en Informática Aplicada
Facultad de Ciencias Exactas – Universidad Nacional de Salta
{ esanchez , daaf }@cidia.unsa.edu.ar

Resumen: El presente trabajo desarrolla un estudio comparativo que evalúa la eficiencia operativa y técnica del protocolo gNMI frente a SNMP en sistemas de gestión de redes. El análisis se estructura en tres dimensiones: funcionalidad del protocolo, modelo de datos semántico y precisión temporal de métricas. Las pruebas se ejecutaron en topologías virtualizadas con Containerlab, generando tráfico ICMPv6, UDP mediante iperf y simulación de ataques de inundación ICMPv6 Neighbor Advertisement. Se midieron métricas de consumo de CPU, jitter temporal, precisión del operador rate() y capacidad de detección de anomalías. El enfoque metodológico permite contrastar el modelo de polling stateless de SNMP contra la arquitectura de streaming stateful de gNMI, evaluando su impacto en la observabilidad y la capacidad de detección de ataques y anomalías de seguridad en entornos de red.

Palabras claves*:* Telemetría, SNMP, gNMI.

1. Introducción

En la última década, la infraestructura de red ha experimentado una transformación profunda impulsada por la adopción masiva de la computación en la nube, la virtualización de funciones de red (NFV) y la proliferación de dispositivos IoT. En este contexto, la red ha evolucionado de una capa de transporte estática a un ecosistema dinámico y crítico en el que la calidad del servicio, la detección de fallos y la planificación de capacidad dependen directamente de la granularidad y la disponibilidad de los datos operativos.

1.1. Las limitaciones del modelo tradicional: tres décadas de SNMP

Durante más de tres décadas, el protocolo SNMP, (Simple Network Management Protocol) ha sido el estándar de facto para la gestión de redes. SNMPv1 fue especificado originalmente en el RFC 1067 [1], (1988) y posteriormente estandarizado en el RFC 1157 [2]. Su versión operativa actual, SNMPv2c, está definida en el RFC 1901 [3] y RFC 3416 [4], mientras que el lenguaje de modelado de datos que utiliza, SMIv2, está especificado en el RFC 2578 [5].1

1 SNMPv3, especificado en RFC 3410 y el framework de RFCs 3411–3418, introduce mecanismos de autenticación y cifrado mediante USM y VACM, pero no modifica el modelo de datos SMIv2 ni el mecanismo de sondeo periódico. Las limitaciones estructurales analizadas en este trabajo son comunes a las tres versiones del protocolo. El presente estudio utiliza SNMPv2c como versión de referencia por ser la más ampliamente desplegada en entornos de producción.

SNMP fue concebido para una era de dispositivos con recursos limitados y topologías relativamente estáticas y predecibles. Su arquitectura se basa en un modelo de sondeo periódico (*polling*) en el que un host de gestión emite solicitudes explícitas, (*GetRequest, GetNextRequest o GetBulkRequest*) a intervalos fijos para recuperar datos operativos de los dispositivos gestionados. Este diseño introduce tres limitaciones estructurales que se agravan a medida que las redes escalan.

En primer lugar, el sondeo genera tráfico de gestión constante independientemente de si el estado de la red ha cambiado, consumiendo ancho de banda y ciclos de CPU del plano de control del dispositivo de forma proporcional al número de objetos monitorizados y a la frecuencia de sondeo. En segundo lugar, el modelo de recolección basado en intervalos crea un punto ciego temporal entre ciclos de sondeo sucesivos, durante el cual pueden producirse y recuperarse eventos transitorios como *micro-bursts*, breves caídas de enlace o errores de corta duración sin ser capturados. En tercer lugar, el modelo de datos SMIv2 asigna identificadores enteros de instancia (*ifIndex*) a los objetos de interfaz, lo que obliga al colector a realizar un *walk* de descubrimiento previo de *ifTable* o *ifXTable* (RFC 2863) [6] para resolver la correspondencia entre nombres de interfaz y sus índices numéricos antes de poder contextualizar las métricas operativas.

1.2. La necesidad de una gestión eficiente y la observabilidad basada en telemetría

La escala y la complejidad operativa de las redes modernas exigen un enfoque fundamentalmente diferente para la recolección de datos. El protocolo gNMI (*gRPC Network Management Interface*), especificado en OpenConfig [7], propone un cambio del paradigma de sondeo, hacia la telemetría de streaming. A diferencia del modelo SNMP, gNMI define un modo de suscripción sobre el modelo de datos YANG (RFC 7950) [8] donde el propio dispositivo empuja datos operativos al colector con precisión temporal, granularidad configurable, serializaciòn eficiente de los datos utilizando Protocol Buffers (*Protobuf*) y mecanismos nativos de seguridad y autenticación, lo que se traduce en mejoras significativas comparado con el modelo de gestión basado en SNMP.

El objetivo del presente trabajo es presentar un estudio comparativo que evalúe la eficiencia operativa y técnica de gNMI frente a SNMP en tres dimensiones: funcionalidad del protocolo, semántica del modelo de datos y precisión temporal en el proceso de recolección de métricas de cada protocolo. El estudio analiza además en qué medida el ecosistema actual de dispositivos está suficientemente maduro para soportar una transición desde SNMP hacia telemetría de streaming, y bajo qué condiciones dicha transición está técnicamente justificada.

2. Metodología adoptada

La presente investigación adopta un enfoque empírico y comparativo para evaluar la eficiencia operativa y técnica de gNMI frente a SNMP. El estudio se estructuró en tres dimensiones analíticas ejecutadas sobre entornos de red virtualizados y controlados, reproducibles mediante la herramienta Containerlab.

2.1. Dimensiones del Análisis

El estudio aborda tres dimensiones que en conjunto permiten evaluar la eficiencia de una solución de telemetría desde perspectivas complementarias: protocolo, modelo de datos y representación temporal.

- **Análisis funcional del protocolo:** Se compararon los mecanismos de transporte, temporalidad y gestión de sesión de ambos protocolos. Por un lado, el modelo de sondeo periódico (*stateless pull*) de SNMPv2c sobre UDP (RFC 3416), en el que el colector inicia cada transacción de forma independiente sin estado de sesión persistente. Por otro, el modelo de suscripción con streaming (*stateful push*) de gNMI sobre gRPC/HTTP/2, en el que el dispositivo mantiene una sesión TCP persistente y empuja actualizaciones al colector según el modo de suscripción configurado (*sample, on_change o target_defined*).
- **Análisis Semántico y Modelo de Datos:** Se examinó la estructura del dato en cada protocolo. En SNMP, la organización tabular de objetos mediante OIDs definidos en SMIv2 (RFC 2578) y el proceso de resolución de índices requerido por IF-MIB (RFC 2863). En gNMI, el modelado jerárquico mediante YANG (RFC 7950), la separación formal entre nodos de configuración (*config true*) y estado (*config false*), y los mecanismos de extensión *augment y deviation*. Se evaluó la capacidad de cada modelo para entregar datos contextualizados sin procesamiento adicional en el colector.
- **Análisis de precisión temporal y fidelidad de la métrica:** Se evaluó cómo el modelo de recolección de cada protocolo afecta la representación temporal de las métricas en el sistema de visualización, con impacto directo en la capacidad de detección de anomalías. Se consideraron como aspectos influyentes, origen de la marca de tiempo, consecuencias en el operador *rate()* de Prometheus e implicancias para la observabilidad operativa.

2.2. Arquitectura de los escenarios de Red

Se configuraron dos escenarios de red virtualizados mediante Containerlab. En ambos escenarios se desplegó un stack de observabilidad compuesto por Prometheus y Victoria Metrics como base de datos de series temporales (TSDB) y Grafana para la visualización de métricas, garantizando una capa de observabilidad uniforme que permite la comparación directa de resultados entre escenarios.

- **Escenario A - Monitorización basada en SNMP:** Se implementó SNMP Exporter [9] como componente de traducción entre SNMPv2c y el formato de exposición de métricas de Prometheus. Se utilizó el archivo snmp.yml por defecto, sin modificaciones al módulo de recolección, configurando únicamente el archivo prometheus.yml para definir el *job de scrape*, el *scrape_interval* y el *target* apuntando al dispositivo monitorizado. El exporter realiza consultas *GetBulkRequest* periódicas al dispositivo y expone los resultados en el endpoint HTTP /metrics, desde donde Prometheus los ingiere en cada ciclo de scrape. [10]
- **Escenario B - Telemetría de red con gNMIc:** Se desplegó gNMIc [11] como colector de telemetría de streaming. Se configuró una suscripción de tipo *stream* con modo *sample* e intervalo de muestreo equivalente al *scrape_interval* del Escenario A, permitiendo la comparación directa del volumen de tráfico bajo condiciones temporales idénticas. Los datos recolectados se enviaron a Victoria Metrics mediante el mecanismo *remote_write* de gNMIc. El dispositivo monitorizado en ambos escenarios fue un nodo

Nokia SR Linux, accedido vía SNMPv2c en el Escenario A y vía gNMI sobre gRPC en el Escenario B. [12]

Adicionalmente se consideró el análisis de funcionalidad de la alternativa basada en Zabbix con base en la revisión bibliográfica de la documentación técnica oficial [13][14], con el objetivo de contextualizar las optimizaciones que introduce sobre el modelo de sondeo SNMP base walk cacheado en memoria, scheduling configurable por ítem y agrupación de OIDs mediante *GetBulkRequest* frente a la arquitectura de SNMP Exporter.

3. Análisis del flujo de trabajo SNMP Exporter

La alternativa SNMP Exporter (Prometheus) se basa en un enfoque “infraestructura como còdigo”, permite una gestiòn descentralizada y basada en archivos. La configuración para las suscripciones a las métricas basadas en MIBs se define en los archivos snmp.yml y prometheus.yml. La siguiente figura muestra el flujo de trabajo para la monitorización de un dispositivo de red.

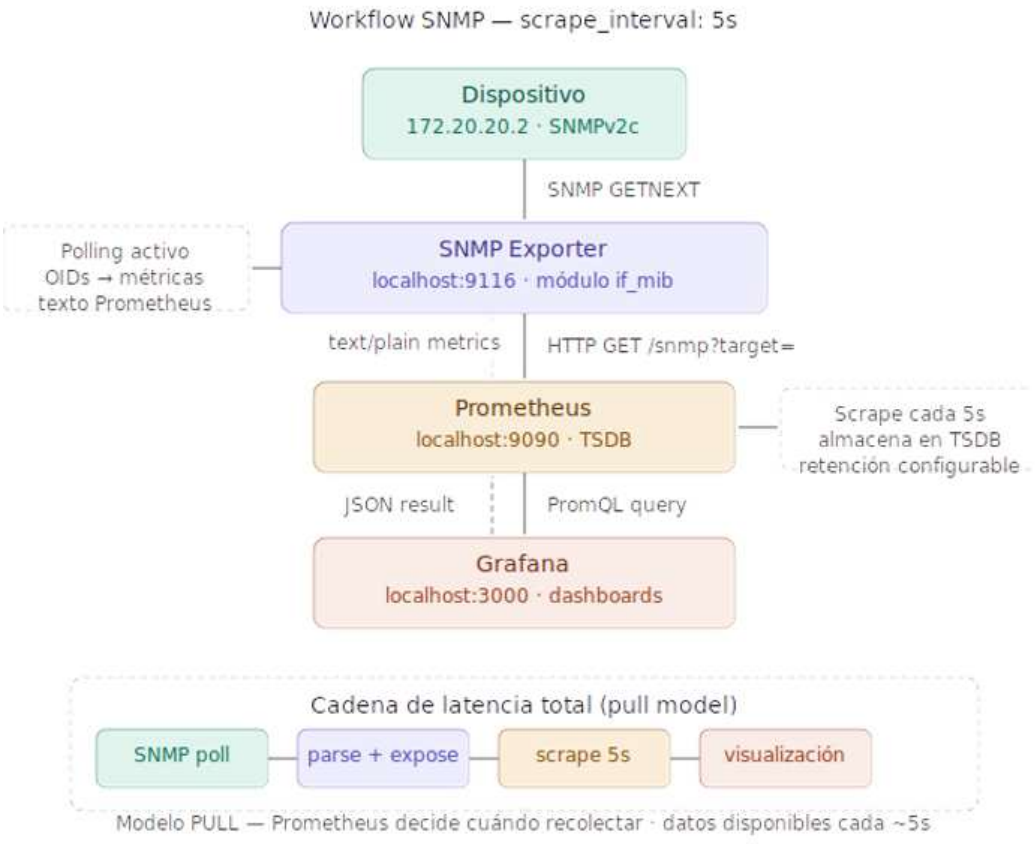

Figura 1: Workflow SNMP Exporter

El workflow opera en dos niveles de iniciativa. En el nivel de observabilidad, Prometheus emplea un modelo pull; es el propio servidor quien decide cuándo recolectar métricas, emitiendo una solicitud HTTP al endpoint /snmp del SNMP Exporter según el *scrape_interval* configurado.

En el nivel de protocolo de red, el SNMP Exporter emplea polling: al recibir la solicitud de Prometheus, emite una consulta *GetBulkRequest* al dispositivo monitorizado y aguarda su respuesta antes de exponer los datos. Ambos mecanismos son iniciados por el colector, nunca por el dispositivo. En base al principio de operaciòn de SNMP (*polling*), se observan tres latencias encadenadas, Prometheus dispara el *scrape*, el SNMP Exporter hace el *GETNEXT walk* al dispositivo, parsea la respuesta y recién entonces Prometheus almacena. Si el dispositivo es lento respondiendo SNMP, toda la cadena se retrasa.

4. Análisis del flujo de trabajo gNMIc Colector

La alternativa basada en la herramienta gNMIc, permite la suscripción a métricas mediante la definición de un archivo de configuración .yml en el cual se definen parámetros generales para la conexiòn al dispositivo a gestionar, las suscripciones a métricas según los modos, (*ONCE, STREAM o POLL*), donde *STREAM* puede a su vez ser (*ON_CHANGE, SAMPLE o TARGET_DEFINED*)[15], se define también el path según el modelo YANG, (componente, contadores, etc) y la salida (destino de los datos recolectados), por ejemplo Victoria Metrics. La siguiente figura muestra el workflow para gNMI.

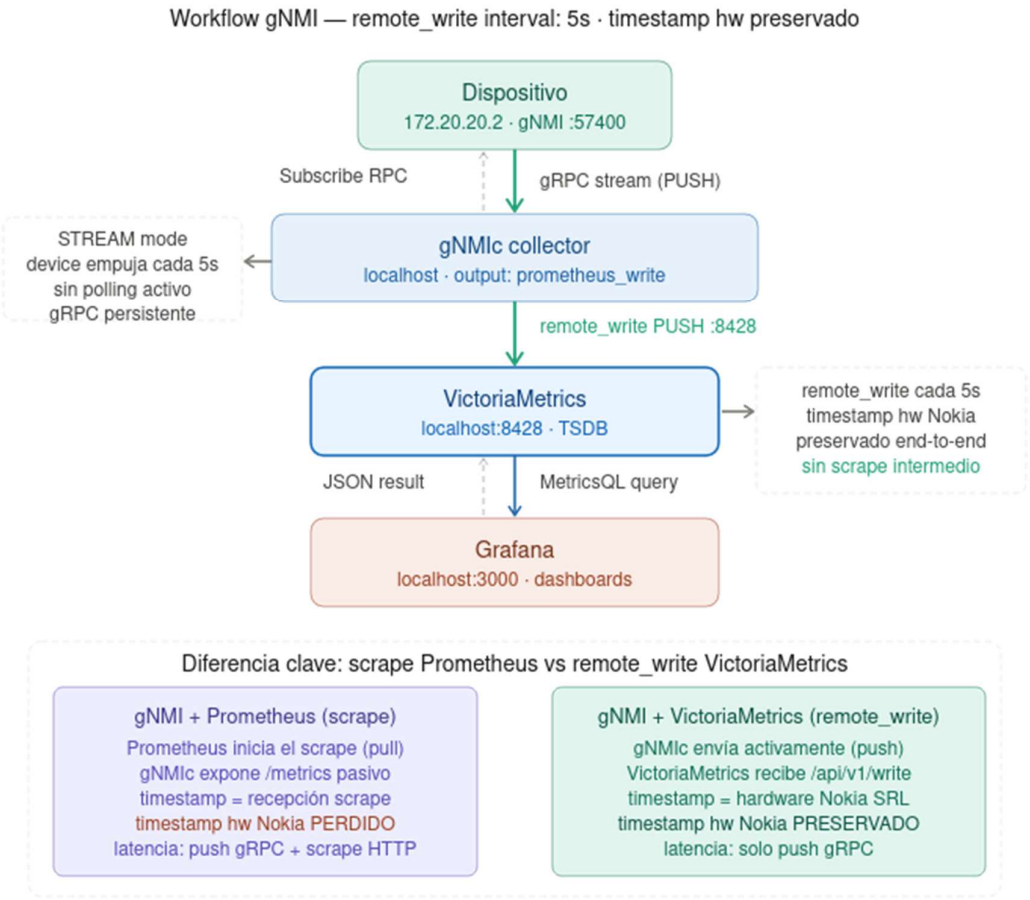

Figura 2: Workflow gNMIc Colector

La primera observación en cuanto a la eficiencia de gNMI frente a SNMP se presenta en el modo de operación, gNMI opera en modo *PUSH/STREAM*, el dispositivo monitorizado, abre una sesión gRPC persistente y empuja telemetría con datos estructurados según el intervalo de tiempo, (*sample-interval*) definido en el archivo de configuración .yml. El flujo se completa con un nuevo *PUSH,* (modo remote write), desde el nodo colector hacia el sistema de monitorización, por ejemplo Victoria Metrics, garantizando una precisión temporal en las métricas subscriptas

Cabe destacar también que los modos de suscripción y stream en gNMI, presentan una optimización en la cantidad de mensajes enviados, donde el dispositivo de red solo envía datos ante actualizaciones. La siguiente figura muestra en detalle los modos de suscripción de SNMP frente a gNMI.

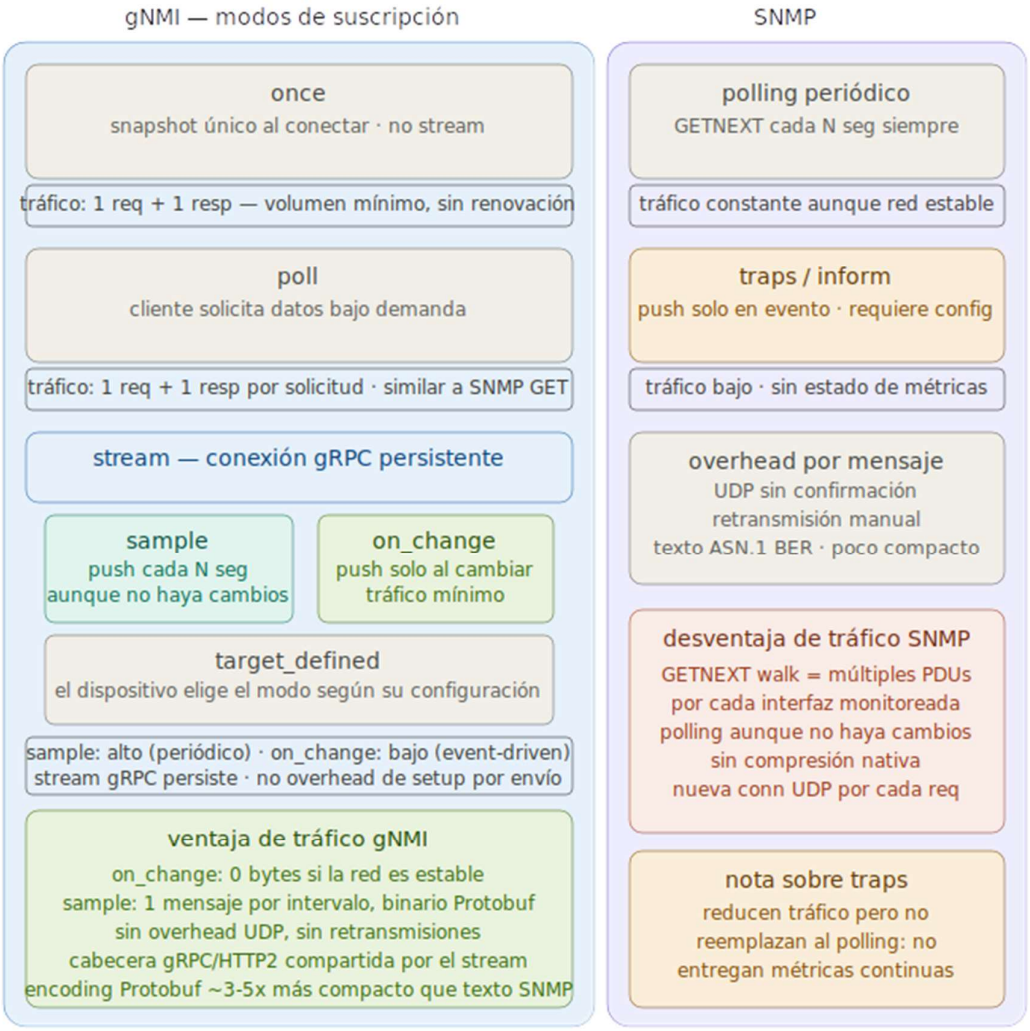

Figura 3: Comparativa Modos de suscripción SNMP vs gNMI

5. Análisis del flujo de trabajo SNMP Zabbix

Zabbix opera con una arquitectura fundamentalmente diferente al stack Prometheus + SNMP Exporter. En lugar del modelo en el que Prometheus decide cuándo recolectar mediante scrape HTTP, Zabbix dispone de un motor de scheduling interno que gestiona cada ítem de monitorización de forma independiente, con intervalos configurables por ítem mediante el parámetro *Update interval* en la definición del ítem o del template. El servidor Zabbix o un proxy Zabbix en arquitecturas distribuidas mantiene una base de datos histórica propia (PostgreSQL, MySQL o TimescaleDB) y expone los datos a Grafana mediante el plugin oficial de Zabbix, o directamente a través de su propia interfaz web sin requerir Prometheus como intermediario.

Para la monitorización SNMP, Zabbix utiliza procesos poller dedicados configurables mediante el parámetro *StartSNMPTrapper* en zabbix_server.conf que ejecutan queries SNMP directamente contra los dispositivos gestionados sin requerir un exporter externo. La optimización principal respecto a SNMP Exporter reside en el mecanismo de *bulk walk;* mediante *GetBulkRequest*, Zabbix agrupa los OIDs pertenecientes a la misma tabla SNMP en una única operación, recupera los valores y los almacena en el value cache en memoria, evitando repetir el *walk* por cada ítem individual en el mismo ciclo de polling.

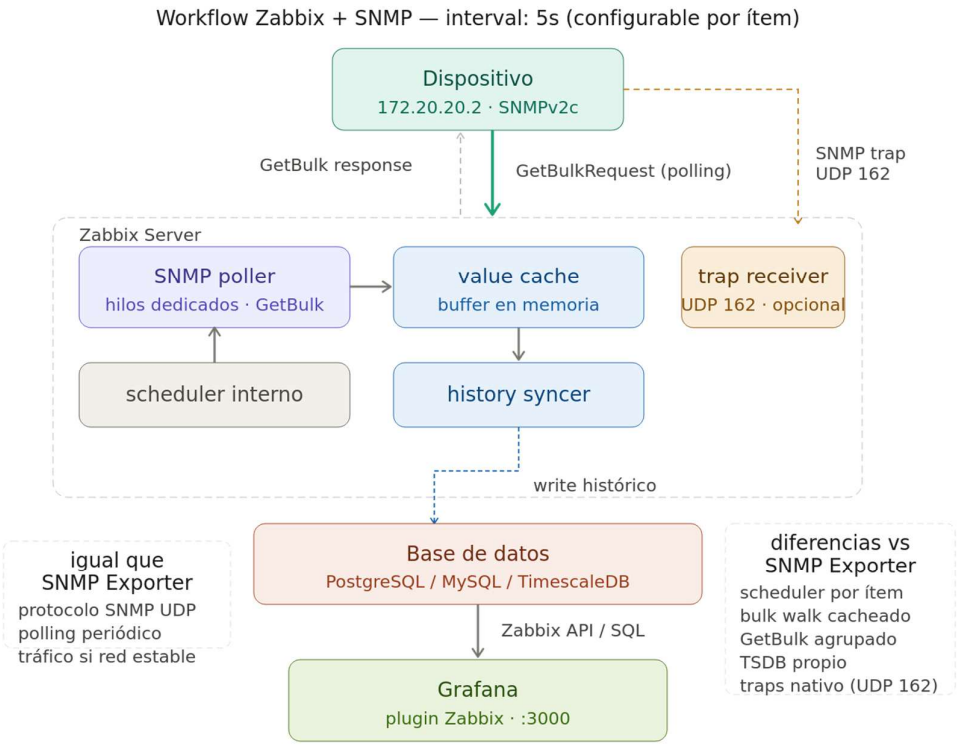

Figura 4: Workflow SNMP Zabbix

Adicionalmente, Zabbix soporta SNMP traps de forma nativa mediante el proceso snmptrapd integrado con el servidor Zabbix, que escucha en el puerto UDP 162. Los traps se configuran como ítems de tipo SNMP trap en el template correspondiente, permitiendo recibir notificaciones asíncronas del dispositivo sin polling activo para ese evento específico.La siguiente figura muestra el flujo de trabajo de esta arquitectura. La optimización de Zabbix sobre SNMP Exporter es real pero estructuralmente limitada, ya que ambos operan bajo el mismo paradigma; polling periódico sobre UDP con encoding ASN.1 BER (RFC 3416, RFC 2578). Las mejoras concretas son tres.

- *Bulk walk* cacheado: SNMP Exporter ejecuta el *walk* completo en cada ciclo de *scrape* de Prometheus, mientras Zabbix recupera los valores de la misma tabla SNMP en una única operación *GetBulkRequest* y los sirve desde el *value cache* en memoria hasta el próximo ciclo. Esto reduce el número de PDUs intercambiados en configuraciones con múltiples interfaces o contadores.
- La agrupación por template permite que los OIDs del mismo host y la misma tabla SNMP se consoliden automáticamente en el menor número posible de *GetBulkRequest,* mientras que SNMP Exporter depende de la definición explícita del módulo en snmp.yml para lograr un comportamiento equivalente.
- El scheduling por ítem permite asignar intervalos de actualización distintos a diferentes contadores, por ejemplo, *ifOperStatus* cada 30s e *ifHCInOctets* cada 5s evitando que todos los objetos monitorizados generen tráfico al mismo ritmo. SNMP Exporter aplica un único *scrape_interval* uniforme a todos los OIDs del módulo.

No obstante, ninguna de estas optimizaciones modifica el paradigma subyacente; Zabbix continúa emitiendo queries SNMP periódicas independientemente de si el estado de la red ha cambiado, exactamente igual que SNMP Exporter. En condiciones de red estable, el volumen de tráfico generado es equivalente al de cualquier solución basada en polling SNMP.

5.1. Complejidad operativa inherente a la arquitectura Zabbix:

A diferencia de SNMP Exporter, cuya configuración se limita a los archivos snmp.yml y prometheus.yml, Zabbix introduce una complejidad operativa estructuralmente mayor. El servidor requiere el despliegue y mantenimiento de una base de datos relacional (PostgreSQL, MySQL) o de series temporales (TimescaleDB), cuya disponibilidad es un requisito crítico: si la base de datos no está disponible, el servidor Zabbix no puede almacenar datos históricos ni evaluar triggers. La configuración de la monitorización se realiza mediante templates conjuntos de ítems, triggers, gráficos y reglas de descubrimiento que deben ser definidos, versionados y mantenidos para cada tipo de dispositivo. Si bien Zabbix provee templates oficiales para equipos de red comunes, la adaptación a dispositivos específicos o a MIBs propietarias requiere la creación o modificación manual de templates, con conocimiento de la estructura de OIDs del fabricante. Esta dependencia de templates específicos por vendor, sumada a la gestión del ciclo de vida de la base de datos, representa una carga operativa significativamente superior a la de un stack basado en SNMP Exporter + Prometheus, y debe considerarse en la evaluación comparativa de viabilidad operativa.

6. Análisis Semántico y Modelado de Datos: MIB vs. YANG

Para evaluar la eficiencia de un protocolo de telemetría es importante comprender la riqueza semántica de su estructura de datos. La forma en que un dispositivo organiza su información interna determina la complejidad del colector y la precisión de la observabilidad. Mientras que SNMP se basa en un modelo de gestión definido en SMI (Structure of Management Information), diseñado para la simplicidad operativa de los años 90, gNMI utiliza modelos YANG, lo que permite una descripción más granular y contextual del estado operativo del dispositivo.

Comprender esta diferencia es fundamental, ya que impacta directamente en cómo los datos son ingeridos, normalizados y correlacionados en plataformas como Prometheus y Grafana. La siguiente clasificación muestra diferencias clave entre ambos modelos.

a. Estructura: Linealidad vs Jerarquía:

- SNMP (SMI/MIB): Aunque el espacio de nombres de objetos se organiza como un árbol OID definido en SMIv2, el acceso a los datos en tiempo de ejecución es esencialmente tabular. Para recuperar información, el colector debe conocer el *Object Identifier* numérico completo (ej. 1.3.6.1.2.1.31.1.1.1.6). La relación entre una interfaz y sus métricas es plana; el identificador de instancia (*ifIndex*) es simplemente un sufijo entero al final del OID base sin expresar semántica sobre el componente al que pertenece.
- Modelo YANG: YANG es intrínsecamente jerárquico. Los datos se organizan en contenedores (container), listas (list) e instancias identificadas por claves (key). Un path gNMI como */interfaces/interface[name='ethernet-1/x']/state/counters/in-octets* describe una ruta lógica y anidada donde cada segmento tiene significado semántico, facilitando la navegación y comprensión sin necesidad de conocer índices numéricos arbitrarios.

b. Riqueza de Datos y Mapeo

- SNMP: La asociación entre una métrica y el componente físico al que pertenece requiere un proceso de descubrimiento previo. Dado que SNMP utiliza índices enteros arbitrarios (*ifIndex*, definido en IF-MIB) para identificar interfaces, el colector no puede solicitar directamente los contadores para una interface especifica. Debe primero recorrer la *ifTable o ifXTable* para construir la tabla de correspondencia entre nombres de interfaz y sus *ifIndex* respectivos. Solo tras este descubrimiento y el posterior cruce de datos en el colector es posible asociar métricas con el componente real. Si bien *GetBulkRequest* optimiza el transporte agrupando múltiples instancias en un único PDU, no elimina la dependencia del *walk* de descubrimiento ni la necesidad de resolución de índices en el colector.
- gNMI**:** Permite suscripción directa al path exacto del modelo YANG. Dado que el nombre de la interfaz es la clave de la lista (list key) en el modelo, por ejemplo */interfaces/interface[name='ethernet-1/x']/state/counters/in-octets*, no existe fase de descubrimiento previa para resolver índices. El colector recibe datos ya contextualizados; el path mismo identifica unívocamente el componente. Esto elimina tanto el *walk* de descubrimiento como la lógica de resolución de índices en el colector, reduciendo su complejidad computacional.

c. Contexto: Implícito vs. Explícito

- SNMP (contexto implícito): En SMIv2, el significado de un objeto está definido en el módulo MIB correspondiente. Sin ese archivo actuando como diccionario, el valor

4294967295 en el OID 1.3.6.1.2.1.2.2.1.10.1 carece de contexto; no es posible determinar si representa un contador de octetos, un índice o un valor de estado. Esta dependencia del archivo MIB para la interpretación semántica es una limitación estructural del modelo. Adicionalmente, SNMPv2c no define un mecanismo nativo para operaciones de configuración complejas; las operaciones *SetRequest* están presentes en el protocolo pero son limitadas en expresividad y prácticamente en desuso en entornos de producción modernos.

- gNMI (contexto explícito): YANG no solo define nodos de datos sino también operaciones mediante RPC y mecanismos de notificación mediante *notification*. Esto permite que el modelo describa acciones con semántica precisa sobre el dispositivo. El path gNMI es autoportante donde el path */interfaces/interface[name='ethernet-1/x']/state/counters/in-octets* identifica unívocamente el dato, su posición en la jerarquía y su relación con el componente padre, sin requerir un diccionario externo para su interpretación.

d. Separación de Datos (Configuración vs. Estado)

- SNMP: SMIv2 no establece una distinción formal entre objetos de configuración y objetos de estado en el espacio de OIDs. Ambos tipos coexisten en la misma tabla, por ejemplo, *ifAdminStatus* (intención del administrador) e *ifOperStatus* (estado real) comparten la *ifTable* sin una separación estructural en el modelo. Esta ausencia de separación formal dificulta determinar solo a partir del OID, si un valor representa un parámetro configurado o una métrica operativa, requiriendo conocimiento previo del módulo MIB para distinguirlos.
- YANG/gNMI: EL RFC 7950 define formalmente los nodos *config true* y *config false*. Los nodos con config true representan la configuración deseada (intención del operador), mientras que los nodos con config false representan el estado operativo real. Esta distinción estructural reforzada por el modelo de datastores de NMDA (Network Management Datastore Architecture, RFC 8342) [16] permite comparar directamente la configuración aplicada contra el estado observado, habilitando la detección de desviaciones entre intención y realidad de forma inmediata y sin procesamiento adicional en el colector.

e. Espacio de Nombres y Extensibilidad

- SNMP: La extensión del modelo de gestión SNMP requiere la definición de nuevos módulos MIB para métricas estandarizadas, esto implica un proceso de publicación a través del IETF, históricamente lento. Como alternativa, los fabricantes definen MIBs enterprise bajo su número de empresa asignado por IANA bajo la rama 1.3.6.1.4.1, lo que genera ecosistemas propietarios fragmentados. El administrador debe gestionar y cargar los archivos MIB de cada fabricante, ante la ausencia de un mecanismo de extensión de modelos existentes implica que las métricas propietarias son estructuralmente independientes de los modelos estándar, dificultando la normalización en entornos multi-vendor.
- YANG: El RFC 7950 define dos mecanismos formales de extensión; *augment*, que permite añadir nodos a un módulo existente sin modificarlo, y *deviation*, que permite declarar diferencias de implementación respecto a un modelo estándar. Los espacios de nombres XML (xmlns) evitan colisiones entre módulos de distintos orígenes. Este diseño permite a los fabricantes extender modelos estándar como OpenConfig o los modelos IETF (ej. ietf-interfaces, RFC 8343) [17] añadiendo métricas propietarias como ramas adicionales, sin invalidar la estructura base ni romper la compatibilidad con colectores

que consumen el modelo original. El resultado es un ecosistema donde los modelos estándar y propietarios coexisten de forma coherente dentro de la misma jerarquía semántica.

El análisis comparativo entre SMIv2/SNMP y YANG/gNMI revela que la diferencia fundamental no es únicamente de protocolo de transporte, sino de modelo epistemológico: cómo cada tecnología representa y comunica el conocimiento sobre el estado de la red.

SNMP, diseñado bajo los principios de SMIv2, prioriza la universalidad y la simplicidad de implementación en el agente. Esta decisión de diseño, acertada para la infraestructura de los años 90, transfiere la carga semántica al colector; la resolución de índices, el walk de descubrimiento, la carga de archivos MIB y la reconstrucción del contexto son operaciones que ocurren fuera del dispositivo, consumiendo ciclos de CPU y memoria en el sistema de monitorización. El dato llega descontextualizado y debe ser enriquecido antes de ser útil.

En contraste, el modelo YANG sitúa la semántica en el origen. El path gNMI es autoportante; identifica el componente, su posición jerárquica y la naturaleza del dato sin procesamiento adicional. La separación formal entre nodos config true y config false, reforzada por NMDA, permite comparar intención y realidad de forma estructural. Los mecanismos de extensión augment y deviation garantizan que las capacidades propietarias de cada fabricante se integren coherentemente en el modelo estándar, sin fragmentar el ecosistema del colector.

Con gNMI, el tiempo del colector se invierte en el análisis de los datos en lugar de en su traducción, acelerando la detección de anomalías y la correlación de incidentes. En el modo on_change, esta eficiencia alcanza su expresión máxima: cuando la red es estable, el tráfico de telemetría es literalmente cero, algo estructuralmente imposible en cualquier variante de SNMP polling.

No obstante, el análisis debe reconocer los límites del contexto. gNMI requiere soporte explícito en el firmware del dispositivo y familiaridad del equipo operativo con modelos YANG, lo que restringe su adopción en infraestructuras legacy. SNMP, respaldado por décadas de implementaciones y una cobertura de dispositivos prácticamente universal, sigue siendo la única opción viable en entornos heterogéneos con equipamiento anterior a 2015. Zabbix, como se analizó, introduce optimizaciones sobre el modelo SNMP base con walk cacheado, scheduling por ítem y GetBulk agrupado, pero no puede superar la limitación fundamental del protocolo; el polling periódico genera tráfico constante independientemente de la actividad real de la red.

7. Análisis de precisión temporal y fidelidad de la métrica

La precisión temporal disponible en el proceso de recolección de métricas de cada protocolo tiene un efecto directo en su representación y visualización, determinando si una anomalía de red de corta duración aparece como evento real o como ruido en la gráfica.

Para el estudio comparativo se realizaron tres pruebas bajo las siguientes condiciones:

- Selección de tres argumentos relacionados que influyen en la precisión temporal y la fidelidad en la representación visual de las métricas: origen del timestamp, operador rate() y observabilidad.

- Métricas *ifInUcastPkts* / *ifOutUcastPkts* / *ifHCInMulticastPkts* / *ifHCOutMulticastPkts* / *ifInErrorPkts* / *ifInDiscardPkts* / *ifHCInOctets* / *ifHCOutOctets* en el escenario con SNMP Exporter
- Métricas *in-unicast-packets* / *out-unicast-packets* / *in-multicast-packets* / *out-multicast-packets* / *in-error-packets* / *in-discarded-packets* / *traffic-rate in-bps* / *traffic-rate out-bps* en el escenario con gNMIc Colector

Prueba 1: Se generó tráfico ICMPv6 mediante mensajes Echo Request / Echo Reply a una tasa de 1 paquete por segundo, se detallan a continuación las observaciones sobre cada uno de los tres argumentos establecidos:

- **Origen del timestamp:** SNMP no define timestamp absoluto en sus PDUs, por lo que el timestamp que Prometheus asocia a cada muestra es el momento de recepción del scrape HTTP en el colector, que incluye la latencia variable de la query *GetBulkRequest* más el tiempo de procesamiento del SNMP Exporter (parseo ASN.1 BER y construcción de la respuesta HTTP). Por el contrario, gNMI define el campo timestamp como nanosegundos Unix generados por el dispositivo en el momento exacto de la lectura del contador (especificación OpenConfig gNMI) [18], independiente de la latencia de red o del tiempo de procesamiento del colector.
- **Consecuencia en el operador rate():** La variabilidad del timestamp SNMP hace que el intervalo efectivo entre muestras no sea exactamente scrape_interval sino scrape_interval ± (latencia de red + tiempo de procesamiento del colector). El operador rate() de Prometheus es sensible a esta variación; en la prueba realizada con scrape_interval = 5s y una tasa de tráfico de 1 pkt/s, la asincronía entre el ciclo de scrape y la cadencia del ping introduce un delta de ±1 paquete entre scrapes consecutivos, produciendo una oscilación de ±0,2 pkts/s en la tasa calculada equivalente a un ruido de ±20% sobre el valor real, representado en la amplitud de sierra observada en la figura 5. En el caso de gNMI, el timestamp del dispositivo garantiza que el intervalo entre muestras sea exactamente sample_interval, haciendo que rate() converja al valor real de 1,0 pkts/s.
- **Implicancia para la observabilidad operativa:** La forma de las gráficas (sierra vs plana) es una demostración visual directa de la diferencia en precisión temporal entre ambos protocolos. La forma de sierra en el escenario SNMP Exporter indica que el intervalo de muestreo del colector no está sincronizado con la cadencia del evento monitoreado debido a que el timestamp de recepción tiene jitter variable. En contraste, la línea plana en las métricas con gNMI, indican que el timestamp de hardware garantiza que cada muestra representa exactamente el mismo intervalo de tiempo, haciendo que rate() sea preciso independientemente de la cadencia del evento.

Las siguientes figuras muestran las observaciones antes descriptas y las diferencias en la representación del flujo de tráfico en el escenario SNMP Exporter vs el escenario gNMIc colector.

Figura 5: Flujo de tráfico ping6 SNMP exporter

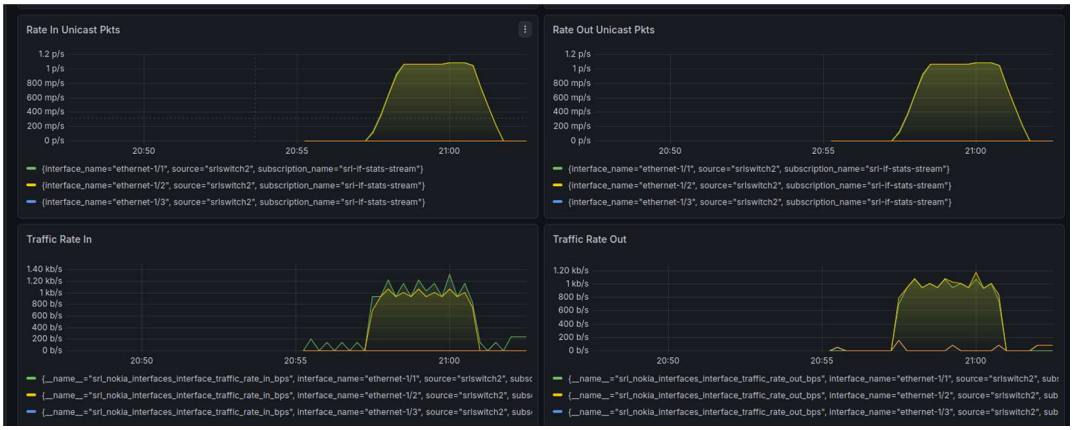

Figura 6: Flujo de tráfico ping6 gNMIc Colector

Prueba 2: Mediante el uso de la utilidad iperf se generó tráfico UDP con una tasa de transferencia de 10Mbps e intervalos de tiempo de 30, 60, 90 y 120 segundos. Se detallan a continuación las observaciones realizadas:

- **Origen del timestamp:** En el escenario basado en SNMP, las métricas de tráfico fueron obtenidas a partir de los contadores acumulativos ifHCInOctets e ifHCOutOctets. Estos contadores únicamente representan valores acumulados de bytes transmitidos y recibidos desde el inicio de operación de la interfaz, sin incorporar información temporal explícita acerca del instante en que ocurrió cada variación de tráfico.
  Como consecuencia, el timestamp finalmente utilizado por Prometheus corresponde al momento del scrap y no al instante real de generación del tráfico en el dispositivo Nokia SR Linux. Para un mismo scrape_interval configurado en 5 segundos para ambos escenarios, el mecanismo SNMP continúa dependiendo de un modelo pull discreto y de la reconstrucción temporal posterior mediante derivación matemática sobre counters acumulativos.
  En contraste, el escenario basado en gNMI la subscripción a métricas traffic-rate in-bps y traffic-rate out-bps, recolectadas mediante gNMIc, representan tasas instantáneas generadas directamente por el dispositivo gestionado y son transmitidas junto con timestamps nativos mediante telemetría streaming.

La diferencia se evidencia claramente en las gráficas. En el escenario gNMI, las métricas presentan una forma trapezoidal, con niveles estables cercanos a los 10 Mbps definidos en iPerf. Esto indica que la tasa observada corresponde de manera directa al comportamiento real del tráfico UDP generado durante intervalos de 30s, 60s, 90s y 120s.
Por el contrario, en el escenario SNMP las curvas presentan una forma triangular. Esta diferencia no está asociada al tráfico generado, sino al mecanismo mediante el cual la tasa debe inferirse a partir de contadores acumulados y timestamps externos. El sistema no observa directamente la tasa de transmisión, sino únicamente diferencias acumuladas entre muestras discretas de bytes.

- **Consecuencia en el operador rate():** La forma triangular que presenta la gráfica en el escenario SNMP Exporter es una consecuencia directa de aplicar rate()[1m] sobre muestras recolectadas con scrape_interval=5s. Cuando el tráfico generado por iperf comienza abruptamente, las primeras muestras de la ventana de 1 minuto aún contienen valores de la fase anterior, por lo que rate() devuelve un promedio ponderado entre cero y el valor real produciendo la rampa de subida triangular. El mismo efecto ocurre en la bajada. La forma de sierra que presenta la gráfica tiene la misma característica ya analizada.en la prueba anterior. Bajo las mismas condiciones de tráfico generado para el escenario con gNMIc, la suscripción traffic-rate/in-bps es un dato calculado directamente por el dispositivo y entregado como valor instantáneo en cada muestra, por lo que no requiere del uso de rate() de Prometheus. Como resultado, el inicio y el fin del tráfico generado son detectados con precisión de exactamente 1 sample_interval (5s).
- **Implicancia para la observabilidad operativa:** Las diferencias observadas tienen implicancias directas sobre la calidad de observabilidad operativa en entornos modernos de monitoreo de redes.
  El enfoque SNMP permite estimar tendencias generales de utilización de ancho de banda, siendo funcional para monitoreo histórico. Sin embargo, debido a que las tasas deben derivarse matemáticamente desde counters acumulativos, la representación temporal del tráfico se encuentra inherentemente desacoplada del comportamiento real de la red. Las formas triangulares observadas en las gráficas constituyen evidencia directa de este fenómeno; el sistema no observa la tasa instantánea real, sino una reconstrucción estadística dependiente de ventanas temporales y frecuencia de polling.
  Por otro lado, el enfoque gNMI basado en métricas de traffic-rate proporciona una representación mucho más cercana al comportamiento efectivo del tráfico generado. Las formas trapezoidales y estables observadas en las gráficas indican que el sistema monitorea directamente la tasa instantánea producida por el dispositivo, preservando tanto la amplitud como la duración temporal real del flujo UDP.
  En términos arquitectónicos, los resultados obtenidos demuestran que la diferencia entre ambos enfoques no depende únicamente de la frecuencia de recolección, sino principalmente del modelo semántico de las métricas disponibles. Mientras SNMP obliga a derivar el throughput desde counters acumulativos, OpenConfig/gNMI expone directamente métricas orientadas a observabilidad operacional de alta fidelidad temporal.

Las siguientes capturas representan los paneles configurados en Grafana para el análisis comparativo antes descripto:

Figura 7: Flujo de tráfico iperf SNMP exporter

Figura 8: Flujo de tráfico iperf gNMIc Colector

Prueba 3: Mediante el uso de la herramienta de auditoría de seguridad THC IPv6 se simuló un ataque de inundación de mensajes ICMPv6 Neighbor Advertise con intervalos de tiempo de 2, 5 y 10 segundos con dos ciclos de ejecución. Para comprender el efecto del ataque y la capacidad de detección en ambos escenarios, se observó en primera instancia el flujo de tráfico ICMPv6 en condiciones normales, y posteriormente se ejecutaron los dos ciclos de ataques, se detallan a continuación las observaciones realizadas:

- **Origen del timestamp:** Durante el ataque flood_advertise6, en el escenario SNMP Exporter, las gráficas observadas en los paneles "Rate In Multicast Pkts" y "Rate Out Multicast Pkts" muestran un patrón escalonado pronunciado con bordes cuadrados donde cada "escalón" representa el intervalo de ataque. La ausencia de una marca de tiempo en origen y la latencia generada por el polling de SNMP y el scrap de Prometheus genera una curva suavizada con mesetas planas. Con gNMIc, el timestamp de hardware garantiza la marca temporal de cada mensaje de telemetría y refleja el instante real de la lectura del contador y no el momento en que el colector lo recibió, esta distinción es opera-

tivamente relevante para correlacionar el inicio del ataque y la detección de ráfagas cortas de tráfico, representadas en la gráfica como picos con contadores más precisos, según se evidencia en los paneles “Rate In Error Pkts” y Rate In Discard Pkts”.

- **Consecuencia en el operador rate():**. La definición correcta de la ventana de tiempo para el operador rate() es determinante para la detección de las ráfagas de mensajes ICMPv6 NA. Con las observaciones realizadas, ambos protocolos permiten la detección de los ataques incluso para el intervalo de 2s, sin embargo en el escenario gNMIc la latencia mínima garantizada por el modo de operación push permite que el operador rate() genere valores con mayor precisión y una representación real del tráfico generado.
- **Implicancia para la observabilidad operativa:** La implicancia operativa emerge de la combinación de los dos argumentos anteriores. Bajo las mismas condiciones, ambos protocolos permiten la detección de los ataques, pero gNMIc permite una observación completa de la “firma” del ataque, no solo detectar el flooding sino también evaluar su severidad, representado en los valores cuantitativos.

Un factor crítico a analizar en este tipo de ataques es el consumo de CPU en el nodo colector, pero fundamentalmente en el dispositivo de red monitorizado. Para tal fin se desplegó un nodo cAdvisor el cual expone las mediciones a Prometheus para la visualización en Grafana. La primera observación relevante es el menor consumo del nodo gNMIc frente al nodo SNMP Exporter, con un factor de eficiencia de 10x aún bajo las condiciones de ataque, lo que claramente refleja la eficiencia del modo de operación push en gNMIc vs el modo poll + scrap en SNMP Exporter.

En relación al dispositivo Nokia Switch, en ambos escenarios de red, el consumo de CPU es idéntico en condiciones de tráfico ICMPv6 normal, donde el cuello de botella se presenta en el plano de control del dispositivo durante el ataque de flood advertise, por lo que podemos concluir que la elección entre SNMP o gNMI no afecta el impacto sobre el dispositivo monitorizado, sin embargo podemos inferir que la eficiencia de gNMI demuestra que permite escalar a entornos de mayor infraestructura. Las siguientes figuras muestras los resultados de las observaciones anteriores.

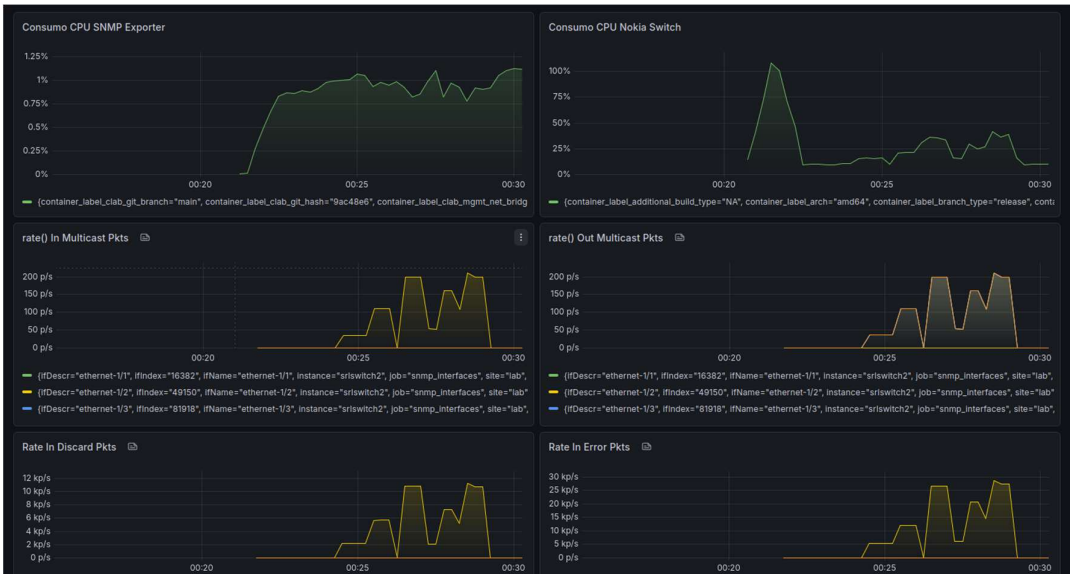

Figura 9: Flujo de tráfico flood advertise attack en SNMP Exporter

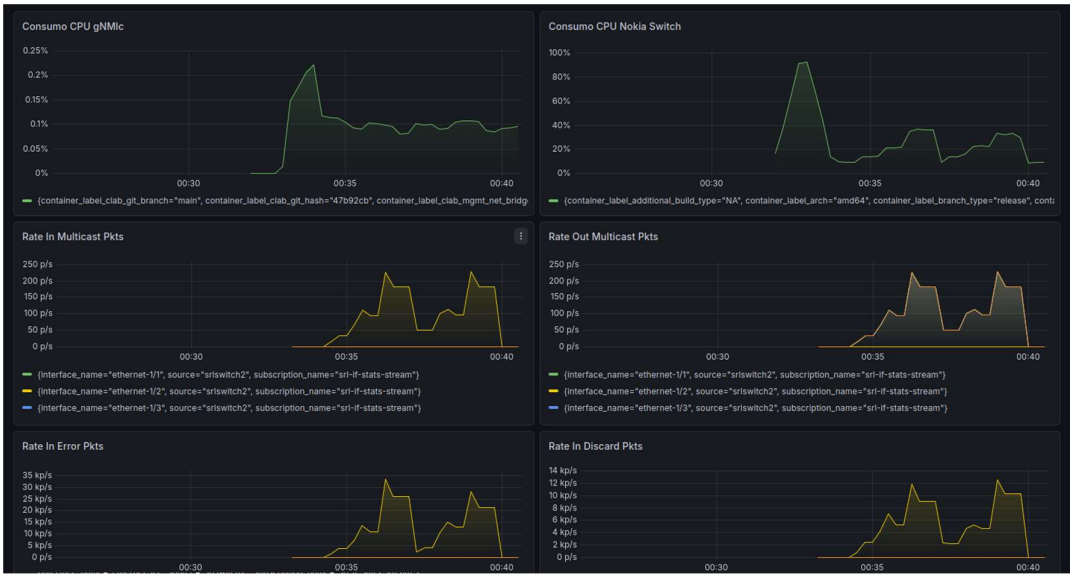

Figura 10: Flujo de tráfico flood advertise attack en gNMIc Colector

8. Conclusiones

El estudio comparativo realizado entre el stack basado en SNMP Exporter y la arquitectura gNMIc demuestra que la telemetría de red ha evolucionado de un modelo de monitoreo reactivo a uno de observabilidad proactiva y determinista. A continuación, se sintetizan los hallazgos en las tres dimensiones analíticas propuestas:

El workflow de SNMP Exporter opera sobre un modelo stateless polling sobre UDP, donde el colector debe interrogar activamente al dispositivo en intervalos fijos, añadiendo una segunda capa de extracción (scrape) por parte de Prometheus. Este esquema introduce latencia inherente, jitter de red y un overhead computacional que escala linealmente con el número de dispositivos y objetos monitorizados. Si bien plataformas como Zabbix mitigan parcialmente estas limitaciones mediante templates, Low-Level Discovery y preprocessing, introducen una complejidad operativa significativa: dependencia de motores de base de datos relacionales, configuración guiada por UI y mantenimiento de MIBs, sin resolver la naturaleza reactiva del polling ni la ausencia de marcas de tiempo nativas.

En contraste, el workflow de gNMIc opera bajo un modelo stateful push mediante suscripciones gNMI sobre gRPC/HTTP2, con TCP como protocolo de transporte, la conexión persistente multiplexada elimina el handshake repetitivo, la serialización binaria (Protobuf) reduce drásticamente el ancho de banda, y el envío continuo vía remote write a VictoriaMetrics garantiza un flujo de datos sin ventanas ciegas. Las pruebas empíricas confirmaron que gNMIc consume aproximadamente 10 veces menos CPU en el colector y escala de manera sub-lineal, transformando la telemetría de una carga operativa a un servicio de infraestructura eficiente.

En relación a la eficiencia en consumo de recursos, tienen implicancia directa la forma en que SNMP estructura la información basada en OIDs definidos en SMIv2 (ej. IF-MIB), donde la recuperación de métricas requiere resolución explícita de índices, lo que obliga al colector a realizar descubrimiento, mapeo y transformación para reconstruir el contexto del objeto monitorizado.

Este enfoque es frágil ante reinicios, reconfiguraciones o cambios de topología, y entrega datos crudos que carecen de semántica intrínseca, desplazando la lógica de negocio al colector.

gNMI, por su parte, se sustenta en modelado jerárquico YANG, donde cada path es auto-descriptivo y contextualizado por diseño. Las etiquetas (labels) se inyectan nativamente durante la suscripción, preservando la relación dispositivo-objeto-métrica, sin procesamiento adicional. Además, las extensiones nativas de fabricantes exponen métricas ya agregadas y semánticamente ricas, eliminando la "tasa de traducción" que SNMP impone al ecosistema de monitoreo. El modelo YANG no solo estandariza la estructura, sino que garantiza que el dato llegue al TSDB con su significado operativo intacto.

La diferencia más crítica radica en el origen de la marca de tiempo y su propagación. SNMP no soporta timestamps nativos; Prometheus asocia a cada muestra un timestamp al momento de recepción del scrape HTTP, que incluye la latencia variable de la query GetBulkRequest más el tiempo de procesamiento del SNMP Exporter. El resultado es un intervalo efectivo entre muestras de scrape_interval ± (latencia de red + tiempo de procesamiento del colector). Esta desincronización obliga al operador rate() de PromQL a extrapolar sobre puntos mal alineados, generando suavizado artificial (evidenciado en la prueba de iperf, donde ráfagas de 30s se subestimaron a ~5 Mbps).

gNMI preserva los timestamps nativos del dispositivo (precisión de nanosegundos), propagándolos sin alteración hasta VictoriaMetrics con remote write. Esto permite que rate() opere sobre una línea de tiempo coherente, eliminando el ruido de extrapolación y habilitando la detección de ráfagas, la correlación causal exacta entre tráfico multicast y descartes, y la reconstrucción forense de ciclos de ataque como se observó en la prueba de ataque de inundación, aportando mayor precisión sobre el efecto del ataque. Operativamente, esto se traduce en umbrales de alerta más sensibles, latencia de detección reducida y una confianza métrica que transforma la observabilidad de "indicativa" a "determinista".

Si bien gNMI se presenta como un estándar técnicamente superior para telemetría moderna, su adopción aún se encuentra en una etapa temprana en el ecosistema de fabricantes, con madurez dispar entre plataformas y brechas en herramientas legacy. Por ello, consideramos conveniente una estrategia de transición basada en un modelo híbrido: manteniendo SNMP para dispositivos heredados y el despliegue de gNMI de forma prioritaria en entornos donde la precisión temporal, la alta cardinalidad o la detección de anomalías en tiempo real sean requisitos de seguridad. Esta estrategia dual permite conservar la inversión existente, validar pipelines de datos en paralelo, y migrar progresivamente dashboards, alertas y automatizaciones hacia gNMI, garantizando continuidad operativa mientras se obtiene el salto cualitativo que la telemetría ofrece a la observabilidad en arquitecturas de redes modernas.

Referencias

[1] https://datatracker.ietf.org/doc/html/rfc1067
[2] https://datatracker.ietf.org/doc/html/rfc1157
[3] https://datatracker.ietf.org/doc/html/rfc1901
[4] https://www.rfc-editor.org/rfc/rfc3416.html
[5] https://datatracker.ietf.org/doc/html/rfc2578
[6] https://www.rfc-editor.org/rfc/rfc2863.html
[7] https://www.openconfig.net/docs/gnmi/gnmi-specification/
[8] https://datatracker.ietf.org/doc/html/rfc7950
[9] https://github.com/prometheus/snmp_exporter

[10] https://github.com/ernestosv73/snmp-exporter.git
[11] https://gnmic.openconfig.net/
[12] https://github.com/ernestosv73/gNMI-comparativa.git
[13] Lontons, A. (2024). *Improving SNMP Monitoring Performance with Bulk SNMP Data Collection*. Zabbix Blog. https://blog.zabbix.com/improving-snmp-monitoring-performance-with-bulk-snmp-data-collection/27231/
[14] Webinar InitMax. SNMP and SNMP traps in Zabbix 7.0. https://www.initmax.cz/wp-content/uploads/2025/01/snmp-a-snmp-traps-in-zabbix-7.0.pdf
[15] https://gnmic.openconfig.net/cmd/subscribe/
[16] https://datatracker.ietf.org/doc/html/rfc8342
[17] https://datatracker.ietf.org/doc/html/rfc8343
[18] https://www.openconfig.net/docs/gnmi/gnmi-specification/#221-timestamps