Telemetry-as-a-Service: Decoupling Collection from Injection for Scalable Network Monitoring

Telemetry-as-a-Service: Decoupling Collection from Injection for Scalable Network Monitoring

Carlos Natalino*, Kaida Kaeval†, Hendrik Johann Kerm†, Torm Järvelill †, Jasper Müller ‡, Paolo Monti*

* Department of Electrical Engineering, Chalmers University of Technology, Gothenburg, Sweden
{carlos.natalino, mpaolo}@chalmers.se

†Tallinn University of Technology, Tallinn, Estonia
{kaida.kaeval, hendrik.kerm, torm.jarvelill}@taltech.ee>

‡Adtran Networks, Martinsried, Germany
jasper.mueller@adtran.com

***Abstract*—Network telemetry has evolved from poll-based SNMP to push-based streaming protocols such as gNMI and YANG-Push, yet current architectures tightly couple device collection with pipeline injection inside monolithic collectors. Scaling telemetry processing under this model requires replicating device credentials across every collector instance, and outsourcing processing to third parties exposes management-plane access together with internal network topology. The term "telemetry-as-a-service" (TaaS) has been introduced in the literature as an API abstraction, but its architecture still combined collection and processing while retaining full device credentials. In this paper we extend the TaaS concept by separating stateful collectors, which face devices and hold credentials, from stateless injectors, which are credential-free, topology-agnostic, and independently scalable. The key enabler is a push-direction inversion: collectors push telemetry to injectors rather than injectors pulling from devices, creating a natural credential boundary. We present a gRPC-based protocol that preserves gNMI semantics while enforcing this boundary, and a proof-of-concept implementation showing that the additional architectural boundary introduces negligible latency and throughput overhead relative to direct gNMI injection. By design, the architecture enables independent horizontal scaling of collection and injection tiers, credential isolation suitable for outsourcing, and topology privacy preservation.**

***Index Terms*—Model-driven telemetry, credential isolation, topology privacy, decoupled aggregation, streaming telemetry.**

I. Introduction

Network telemetry is the foundation of modern network operations. It supports troubleshooting, capacity planning, traffic engineering, and increasingly, closed-loop automation [1]. As networks grow in scale and complexity, operators require sub-second visibility to detect microbursts, identify anomalies, and trigger real-time control actions [2], [3]. Streaming telemetry protocols have emerged to meet this need, replacing the periodic polling model with continuous, push-based data delivery [4], [5]. Meanwhile, the convergence of sensing and communication in wireless [6] and optical networks through state-of-polarization monitoring [7] and optical-spectrum-as-a-service (OSaaS) models [8] is blurring the boundary between user data and monitoring data, further increasing telemetry volumes and the need for scalable processing architectures.

The shift from simple network management protocol (SNMP) to streaming telemetry, with protocols based on gRPC remote procedure call (gRPC), gRPC network management interface (gNMI), and YANG, has been driven by well-documented limitations of the poll-based model. SNMP polling introduces high latency between state changes and their observation, transmits redundant data when values have not changed, and creates inconsistent snapshots when multiple devices are polled sequentially [4], [9]. Model-driven telemetry standards such as gNMI, YANG-Push [10], and OpenConfig address these issues through structured data models and push-based subscriptions that can operate periodically or on-change [11]. In particular, YANG-Push [4] allows devices to negotiate subscription capacity and decline requests that would overload their resources [4].

Despite these protocol advances, the architectures that consume streaming telemetry remain monolithic. Current collectors such as Telegraf and gNMIc combine device connection management, data retrieval, transformation, and pipeline injection within a single process. Scaling processing capacity or ensuring resilience against single points of failure requires replicating the entire collector stack, including device credentials [1]. This tight coupling creates a fundamental problem: the collection of data from devices is substantially different from its storage, processing, and analysis, yet both responsibilities are bound together. Outsourcing telemetry processing to third parties or to a separate network domain exposes device management credentials and internal topology [12]. As network size, sampling frequency, and sample dimensions grow, single collector entities become bottlenecks that cannot scale independently [11].

Prior work has recognized the need for decoupling. NetVision pioneered the "telemetry-as-a-service" terminol-

This work has been supported by the Horizon Europe ECO-eNET project with funding from the SNS JU under grant agreement No. 101139133.

The full implementation and metadata related to this paper are available at https://github.com/carlosnatalino/EuCNC_2026_Telemetry-as-a-Service.

ogy, proposing a service-oriented application programming interface (API) abstraction that separates telemetry applications from infrastructure [13]. However, NetVision's Vantage Server combines probe injection and result collection with full device credentials and complete topology knowledge. It decouples applications from the telemetry mechanism but does not decouple credentials or topological knowledge from processing. The outsourcing problem persists: operators cannot delegate telemetry processing without simultaneously granting device access or direct connection to network devices. True decoupling requires an architectural boundary where credentials and topology knowledge are strictly isolated, with the added benefit that privacy constraints can be enforced through anonymization at this boundary.

In this paper, we present telemetry-as-a-service (TaaS), an architecture that extends the push paradigm beyond device-to-collector communication to collector-to-injector communication. The critical distinction from previous literature is that TaaS injectors possess zero device credentials and zero topology knowledge. Collectors remain stateful, device-specific, and topology-aware, while injectors become stateless and generic. Injectors process opaque telemetry streams without the need (or ability) to identify which devices generated the data or how devices interconnect. This design enables private outsourcing: third-party injectors cannot pivot to device management or infer network structure. The architecture draws inspiration from message broker integration patterns [14] and privacy-preserving telemetry techniques [15], but introduces credential isolation as a first-class architectural property.

The contributions of this paper are:

- A credential-isolated architecture that separates device-aware collectors from credential-free, topology-agnostic injectors, enabling secure outsourcing of telemetry processing.
- A gRPC-based protocol design that preserves gNMI semantics while enforcing a credential boundary through push-direction inversion.
- A stateless injector model that enables elastic horizontal scaling and deployment in untrusted environments [16].
- A quantitative evaluation showing negligible overhead relative to direct gNMI injection and confirming that collector and injector tiers can be scaled independently.

Illustrative results on a proof-of-concept implementation show that the proposal architecture and protocol introduce no meaningful overhead over a traditional gNMI-based architecture.

II. Background

A. Network Telemetry Taxonomy

Network telemetry encompasses three primary data collection approaches. Packet-based collection provides high fidelity at the cost of substantial overhead, flow-based collection offers scalability with coarser granularity, and log-based collection captures event-driven state changes [9]. Orthogonally, active measurement injects synthetic probes to obtain end-to-end metrics, while passive measurement leverages existing counters maintained by network devices [17]. In-band network telemetry (INT), introduced in [18], represents a distinct approach that embeds telemetry metadata directly into data-plane packets, enabling hop-by-hop visibility but facing maximum transmission unit (MTU) constraints and per-switch processing overhead [19], [20].

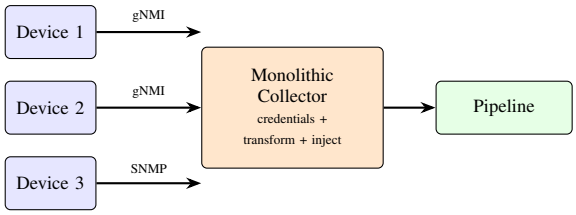

Fig. 1. Traditional monolithic collector architecture. Device credentials, data transformation, and pipeline injection are coupled in a single process, also representing a single point of failure.

B. Streaming Telemetry Protocols

gNMI is a gRPC-based protocol that provides *Get*, *Set*, and *Subscribe* remote procedure calls (RPCs) operating over yet another next generation (YANG) path-based data models [11]. YANG-Push [4] defines a subscription mechanism supporting both periodic and on-change delivery modes. On-change subscriptions minimize network usage by transmitting only modified values, while periodic subscriptions guarantee regular updates at configurable intervals. Devices retain the ability to negotiate subscription parameters and decline requests that exceed their processing capacity [21].

C. Current Architectures and Their Limitations

Fig. 1 illustrates the conventional telemetry architecture. A monolithic collector handles device connection management, subscription lifecycle control, data transformation, and pipeline injection as a single unit. Horizontal scaling replicates the entire stack, including credential management, across every instance. Vertical scaling is limited by single-process constraints on memory and CPU [11]. Message brokers such as Apache Kafka are increasingly used downstream of collectors [14], [22], but the collection stage itself remains tightly coupled.

This coupling produces several operational problems. First, credential sprawl: every collector instance requires device management access, expanding the attack surface proportionally. Second, a compromised collector exposes all connected device credentials [12]. Third, scaling granularity is mismatched: injection throughput cannot be scaled independently of collection capacity. Fourth, outsourcing to cloud-based processing requires sharing device credentials with third parties. These limitations motivate an architecture that cleanly separates the two responsibilities.

III. Related Work

The evolution of network telemetry from static resource polling to dynamic data streaming represents a fundamental shift in how operators perceive network state. Traditional protocols like SNMP relied on a request-response model that

inherently limited visibility due to the processing overhead imposed on network devices [23], [24]. The transition to model-driven telemetry, exemplified by YANG-Push [4] and gNMI, introduced a subscription-based paradigm where devices continuously push state changes to subscribers. This change improved bandwidth efficiency and data granularity by allowing "on-change" updates rather than redundant periodic snapshots [5], [10]. However, while these protocols optimized the device-to-collector link, the collector itself did not change, remaining a monolithic termination point for all state. Our work extends this evolution by applying the push-paradigm to the collector-to-injector link, effectively removing the requirement for the processing layer to maintain any stateful relationship with the data source.

The concept of separating telemetry goals from underlying mechanisms has been explored through the lens of programmability and service abstraction. Yu proposed a "top-down" approach where high-level measurement intents are compiled into device-specific primitives, decoupling the query from the hardware [1]. Similarly, the NetVision architecture introduced the term telemetry-as-a-service (TaaS) to describe an API layer that allows applications to consume telemetry without managing the physical network [13]. While these works successfully decoupled the *application* from the *infrastructure*, they retained a centralized vantage point (e.g., the Vantage Server) that possessed full network privileges. In contrast, our architecture enforces this separation at the topological level, ensuring that the service provision layer (the Injector) is architecturally incapable of accessing the managed infrastructure, rather than simply being policy-restricted.

As networks span multiple administrative domains, the semantic value of telemetry data has shifted from purely operational logs to tradeable digital assets requiring sovereignty. Frameworks incorporating the international data space (IDS) architecture allow operators to share telemetry with external stakeholders while enforcing usage control policies [25]. Jafari et al. extended this to 6G networks with the NOBS framework, identifying telemetry as a sovereign commodity that must be protected by marketplaces and connectors [26]. Kaeval et al. demonstrated the viability of such sharing in optical networks, highlighting the need for robust transport over public networks [8]. However, these solutions often rely on complex overlay connectors to enforce sovereignty. Our contribution complements these frameworks by providing the underlying privacy-focused transport; by physically separating the credential holder from the data distributor, we ensure that data sovereignty is enforced by the network architecture itself.

Privacy in network telemetry has traditionally focused on protecting user traffic or anonymizing specific data fields. Techniques such as oblivious routing and differential privacy have been adapted to decouple the content of a message from the identity of its sender [12]. Zhou et al. proposed using autoencoders to transmit latent representations of telemetry, ensuring that raw data is never exposed to the analysis pipeline [15]. Zafar et al. introduced usage control to prevent analytics functions from processing data outside designated trust zones [27]. While these methods protect the *content*, they do not address concerns related to the *infrastructure*. We address this gap by providing "infrastructure privacy"; our push-inversion model ensures that the analysis layer does not even know the Internet protocol (IP) addresses or topology of the source devices, preventing the inference of network structure.

Finally, the massive scale of modern telemetry data necessitates architectures that can grow elastically without operational friction. Kablan et al. demonstrated that decoupling state from processing is the key to scaling network functions, allowing instances to be added or removed instantly [16]. Woo et al. further refined this by managing state as distributed shared objects to hide access latency [28]. In the monitoring domain, Basin et al. showed that handling out-of-order streams from multiple sources is solvable with watermark-based reordering [29]. Graf et al. proposed a data mesh approach where schema registries allow message brokers to decouple producers from consumers [14]. Drawing on these principles, we apply strict statelessness to the injection tier; this allows our downstream components to scale horizontally like web services, independent of the state-heavy requirements of managing persistent device connections.

TABLE I
POSITIONING OF TAAS AGAINST REPRESENTATIVE PRIOR WORK AND PRODUCTION COLLECTORS ALONG FOUR ARCHITECTURAL PROPERTIES.

System	Cred. iso.	Topo. iso.	Stateless	Push
Telegraf / gNMIc	×	×	×	n/a
NetVision [13]	×	×	×	×
Graf et al. [14]	partial	×	×	✓
Odoh et al. [12]	✓	partial	×	×
TaaS (this work)	✓	✓	✓	✓

Table I summarizes how TaaS differs from representative production collectors and prior architectural proposals along four properties: *credential isolation* (the processing tier never holds device credentials), *topology isolation* (the processing tier is unaware of device identities and their interconnection), *statelessness* of the processing tier, and *push* direction at the collector-to-processing link. No prior system combines all four; TaaS achieves them through the push-direction inversion described in Section V.

IV. TELEMETRY-AS-A-SERVICE (TAAS) ARCHITECTURE

The TaaS architecture is guided by four principles. First, *separation of concerns*: device-facing collection is distinct from pipeline-facing injection [1]. Second, *stateless processing*: injectors maintain no per-device or per-subscription state, enabling elastic scaling without state migration [16]. Third, *protocol compatibility*: the architecture preserves gNMI data structures (*Notification*, *Update*, *TypedValue*) to ensure semantic fidelity. Fourth, *minimal trust*: injectors require no device credentials, no topology knowledge, and no awareness of the collection infrastructure.

Fig. 2 shows the TaaS architecture. The system comprises two tiers separated by a credential boundary. Collectors reside

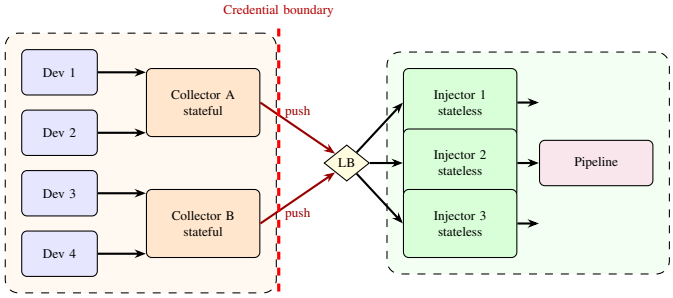

Fig. 2. TaaS architecture. Collectors push telemetry across a credential boundary to stateless injectors, which can be deployed in untrusted environments. The load balancer (LB) distributes push requests across the injector pool.

in a trusted zone alongside managed devices. Injectors operate in a potentially untrusted zone alongside the processing pipeline. The push direction flows from collectors to injectors, inverting the traditional model where processing components pull from or receive connections initiated by devices.

In this architecture, collectors are stateful components that maintain persistent connections to network devices and manage subscription lifecycles. Each collector holds device credentials, understands device capabilities, and can interact with device-specific YANG models or legacy protocols such as SNMP or NETCONF. Their processing is lightweight: they transform device-specific notifications into TaaS messages with minimal computation. This software-based design allows collectors to run on commodity hardware, virtual machines, or even containerized specifically on network equipment, without requiring specialized appliances. Collectors are deployed on-premises or at edge locations close to the managed devices. They scale with the number of devices, and credential partitioning limits the exposure per collector instance. The isolation from external networks allows operators to use legacy protocols with reduced vulnerability exposure.

Injectors are stateless components that accept telemetry from any collector without requiring device knowledge, credentials, or topology awareness [16]. A standard load balancer distributes incoming requests across the injector pool. Injectors validate message structure and credentials, and inject telemetry into the downstream pipeline, which may be a message broker such as Apache Kafka [30], a time-series database, or a stream processing engine. They can be deployed in a cloud environment, a central datacenter, or on third-party infrastructure [14]. Injectors are designed to execute on standard server hardware, relying on horizontal scaling rather than vertical hardware acceleration. Because injectors hold no state, instances can be added or removed without affecting ongoing collection or requiring state migration.

A key advantage of the TaaS architecture is that collector and injector tiers scale independently. When the network has many devices but each produces a low data rate, operators deploy many collectors and few injectors. Conversely, when few devices produce high-rate telemetry, a small number of collectors can feed a large injector pool. Bursty workloads benefit from auto-scaling the injector pool without modifying the collector deployment [31]. This approach is compatible with adaptive monitoring strategies that vary telemetry resolution across time and across metrics. Each tier can therefore be right-sized independently based on its actual bottleneck, avoiding the over-provisioning inherent in monolithic scaling.

The TaaS architecture enforces credential isolation through an architectural boundary. Collectors are trusted with device credentials and deployed in secure network zones with full topology knowledge. Injectors hold zero credentials and zero topology knowledge, making them deployable in untrusted environments [12]. The push direction enforces this boundary structurally: because injectors receive data pushed by collectors, they have no mechanism to request data from devices or to discover device addresses. This contrasts with NetVision, whose Vantage Server requires device access for probe injection [13].

The source identifier in the TaaS protocol is an opaque collector-assigned value, which can be configured not as a device identifier to improve privacy. Injectors see telemetry payloads but not device IP addresses, device hostnames, or the relationships between devices. Collectors can optionally anonymize or aggregate data before pushing, adding a further privacy layer. Even with full access to injector logs, an attacker cannot reconstruct the network topology or identify specific devices.

We frame the privacy analysis of TaaS against an adversary whose goal is either to *(i)* obtain device management credentials, *(ii)* reconstruct the managed network topology, or *(iii)* tamper with telemetry data in transit. We assume mutual transport layer security (TLS) between collector and injector, standard gRPC authentication, and that collector binaries and their credential stores are not compromised. The latter assumption is strictly weaker than in the monolithic case, because credentials are partitioned per collector rather than replicated across all processing nodes. Compromising a collector exposes only the credentials and topological information of the device subset assigned to it. In the monolithic design, the equivalent compromise exposes all device credentials and topology, typically replicated across instances. Compromising an injector or a downstream processing node yields no credentials, no device addresses, and no topological information, only opaque payloads. In the monolithic case, by contrast, the co-located collection stage exposes both credential and topology at the same point. Pipeline consumers retain the same isolation they already enjoy in both designs. The collector $\leftrightarrow$ injector link is protected by gRPC over TLS with sequence numbers preventing replay, matching the gNMI-over-TLS guarantee of traditional telemetry solutions. The residual adversary capability under TaaS is therefore limited to observing opaque telemetry payloads, which matches the threat model of any downstream analytics pipeline.

V. Protocol Design

The TaaS protocol preserves the telemetry semantics of gNMI while adding metadata required for decoupled oper-

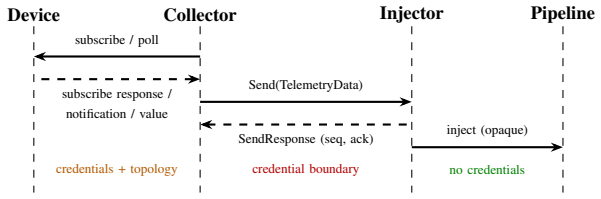

Fig. 3. Message flow from device to pipeline through a collector and a TaaS injector. The credential boundary is crossed only by *TelemetryData* payloads.

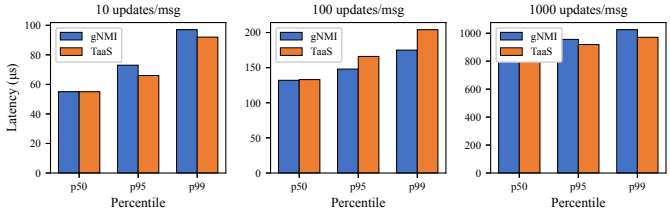

Fig. 4. Latency percentiles (p50, p95, p99) for direct gNMI and TaaS injection at three payload sizes, without concurrency.

ation. The design goals are: preserving timestamps, paths, and typed values from gNMI; minimizing overhead relative to native gNMI message passing; enabling collector identification for debugging and access control; and supporting batching for throughput and sequencing for ordering guarantees.

TaaS *TelemetryData* reuses the gNMI *Notification* as its payload, preserving all original fields: timestamp, prefix, update list, delete list, and the atomic flag. Three additional fields support decoupled operation. The `collection_timestamp` records when the collector received the original *Notification*, enabling end-to-end latency measurement independently of device clock accuracy. The `source_id` identifies the originating collector for tenant routing and debugging, without revealing device identity. The `sequence_number` supports ordering verification and deduplication across the credential boundary.

The TaaS service exposes a single RPC: `Send(TelemetryData)` returns `SendResponse`. This unary request-response pattern prioritizes simplicity and reliability. The gRPC transport provides HTTP/2 multiplexing, TLS encryption, and compatibility with standard load balancers. The `SendResponse` includes an acknowledgment status and the received sequence number for flow control. Future extensions may include streaming RPCs for sustained high-throughput scenarios and bidirectional streams for backpressure signaling.

Fig. 3 illustrates the end-to-end message flow. The architectural enabler of credential isolation is the inversion of the push direction relative to gNMI. In gNMI Subscribe, the client (collector) initiates a connection to the server (device) and requests data, following a pull model where the requester must authenticate to the data source. In TaaS Send, the client (collector) initiates a connection to the server (injector) and pushes data. The injector never initiates connections to devices and has no mechanism to do so. The same underlying data model allows seamless transformation between gNMI *Notification* and *TelemetryData* messages, preserving semantic fidelity while enforcing the credential and privacy boundaries.

VI. Evaluation

We implement the TaaS collector and injector in the Rust programming language, using the `tonic` library for gRPC and `prost` for Protocol Buffer serialization. The baseline is a direct gNMI injection path where a collector serializes *Notification* messages and writes them to the pipeline without an intermediate injector. We compare this baseline against the TaaS path where the collector sends *TelemetryData* messages through an injector before pipeline delivery. The evaluation is protocol-level and independent of the underlying network technology: payloads follow the gNMI *Notification* data model with random *TypedValue* fields generated at configurable payload sizes of 10, 100, and 1000 updates per message, representing low, medium, and high data density scenarios typical of packet, optical, or radio access network telemetry feeds. All components run as separate processes on a single host (Linux, loopback interface) to isolate protocol and serialization overheads from network-transport effects; each configuration is run for 50 000 messages. We measure latency (p50, p95, p99), throughput (messages per second and bytes per second), and central processing unit (CPU) utilization.

A. Latency Analysis

Fig. 4 compares the latency distribution for a single collector-injector pair. At 10 updates per message, both protocols achieve a p50 latency of 55 µs. TaaS is marginally lower at the tail (p95 66 vs. 73 µs, p99 92 vs. 97 µs). At 100 updates per message, p50 is within 1 µs (133 vs. 132 µs) but TaaS is slightly higher at the tail (p95 166 vs. 148 µs p99 204 vs. 175 µs, a 12–17% gap). At 1000 updates per message, TaaS is marginally faster at all percentiles (p50 849 vs. 851 µs, p99 971 vs. 1025 µs). The sign and magnitude of these differences are consistent with measurement variance on a shared single-host testbed rather than a systematic overhead of the added architectural boundary. Across all payload sizes the median differences remain within a few percent, and no systematic degradation is introduced by the TaaS path.

B. Throughput Analysis

Fig. 5 shows throughput at concurrency 10, where both protocols benefit from request pipelining. At 10 updates per message both protocols saturate around 27 000 msg/s; at 1000 updates, TaaS achieves 3 497 msg/s versus gNMI's 5 618 msg/s. The larger relative gap at 1000 updates reflects the cumulative cost of serializing the *TelemetryData* wrapper around a large *Notification* payload. At concurrency 1 the overhead is within measurement variance across all payload sizes.

C. Independent Scaling

To validate that collector and injector tiers scale independently, we run the TaaS path with n_c collector processes and

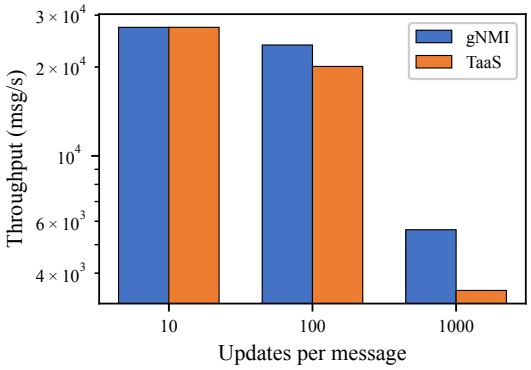

Fig. 5. Throughput comparison in terms of number of value updates per message between direct gNMI and TaaS injection at concurrency 10 (i.e., 10 independent threads).

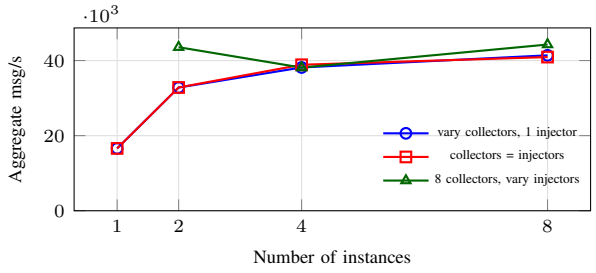

Fig. 6. Independent scaling validation. Aggregate throughput scales sub-linearly with the number of collectors (payload 100 updates, concurrency 10 per client) on a single-host testbed; the number of injectors can be varied independently of collectors without disruption.

n_i injector processes on the same host and measure aggregate throughput at 100 updates/message and concurrency 10 per collector. Fig. 6 shows three scenarios: *(i)* fixing $n_i = 1$ and varying n_c, *(ii)* matching $n_c = n_i$, and *(iii)* fixing $n_c = 8$ and varying n_i. In scenario *(i)-(ii)*, aggregate throughput increases from 16.6 kmsg/s ($n_c = 1$) to 32.8 kmsg/s ($n_c = 2$), then rises more slowly to ≈41 kmsg/s at $n_c = 8$ as the single-host testbed becomes CPU-bound across all processes. In scenario *(iii)*, varying n_i from 2 to 8 while keeping $n_c = 8$ produces fluctuations of ±10% around 40 kmsg/s, confirming that the injector tier is not the bottleneck at this scale and that instances can be added or removed without disrupting ongoing collection. These measurements are protocol-level and under-estimate the scaling headroom achievable in a multi-host deployment where each tier runs on its own hardware; they nevertheless confirm that TaaS exposes collector and injector counts as independent scaling knobs, consistent with the design property claimed in Section IV.

VII. Discussion

A. Deployment Considerations

TaaS supports incremental adoption. Collectors can coexist with legacy monolithic collectors during a migration period, as both can feed the same downstream pipeline. Collector placement at the network edge minimizes wide area network (WAN) bandwidth consumption for telemetry data. Injector placement in cloud-native environments enables elastic auto-scaling through standard container orchestration platforms. Standard gRPC load balancers distribute traffic across the injector pool without requiring application-level routing logic.

Applicability. The protocol is agnostic to the underlying network technology: any device that exposes a gNMI, YANG-Push, NETCONF, or even SNMP interface can be fronted by a collector that transforms the native notifications into *TelemetryData* messages. This makes TaaS directly applicable to packet networks (OpenConfig-based routers), optical networks (OpenConfig-optical, OSaaS monitoring feeds), and radio access networks (O-RAN SMO telemetry), as well as to cross-domain scenarios where different collectors wrap different underlying protocols while sharing a single injector pool.

Fault tolerance and scaling. Because injectors are stateless, fault tolerance reduces to the standard problem of load-balancing a stateless gRPC service: instances can be restarted, added, or removed without affecting in-flight subscriptions. Collectors, in turn, provide at-least-once delivery via the `sequence_number` field, and injector-side deduplication is straightforward when downstream sinks require exactly-once semantics. Auto-scaling policies can target injector CPU and queue depth independently of collector scaling, which is instead driven by the number of managed devices and subscription volume.

Pipeline integration. Injectors are intentionally thin: an injector implementation can forward *TelemetryData* directly into message brokers such as Apache Kafka [14], [30], time-series databases, stream processors, or anomaly-detection frameworks [32]. An adapter injector can also emit the payload in formats consumed by existing toolchains (Telegraf inputs, OpenTelemetry exporters), lowering the adoption barrier without disturbing the credential boundary.

B. Limitations and Future Work

The current protocol uses unary RPCs, which require a round trip for each message. Streaming RPCs would improve throughput for continuous telemetry by reducing connection and handshake overhead across many messages. End-to-end encryption from collector to pipeline consumer is not addressed in the current design, but this can be included using state-of-the-art mechanisms. Collector orchestration and auto-discovery for dynamic device inventories remain open problems. Real-world evaluation with production devices from multiple vendors would validate the architecture under diverse protocol implementations. Integration with existing collector platforms such as Telegraf and gNMIc as TaaS output plugins would lower the adoption barrier.

VIII. Conclusion

Current telemetry architectures couple device collection and pipeline injection, limiting both scalability and the ability to outsource processing. TaaS introduces credential isolation as a first-class architectural property: injectors hold zero credentials

and zero topology knowledge. Collectors remain stateful and credential-holding; injectors become stateless, generic, and safely deployable in untrusted environments. The proposed push-direction inversion creates a natural credential boundary, ensuring that injectors cannot request data from or discover devices.

Our evaluation confirms that the collector-injector boundary adds no measurable latency or throughput overhead relative to direct gNMI injection, and that collector and injector counts can be scaled as independent knobs. This opens deployment models that prior architectures cannot support: third-party telemetry processing without credential exposure, multi-tenant shared infrastructure with tenant isolation, and independent optimization of collection and injection tiers. Future work will explore streaming protocols for continuous telemetry, integration with data privacy techniques at the collector layer, and production deployment studies with network equipment.

References

[1] M. Yu, "Network telemetry: Towards a top-down approach," *SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, pp. 11–17, Feb. 2019. DOI: 10.1145/3314212.3314215.

[2] "Big data driven networking – Requirements and capabilities of network visibility," ITU-T, Recommendation ITU-T Y.3657, Dec. 2023.

[3] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework," Internet Engineering Task Force, Request for Comments RFC 9232, May 2022. DOI: 10.17487/RFC9232.

[4] A. Clemm and E. Voit, "Subscription to YANG Notifications for Datastore Updates," Internet Engineering Task Force, Request for Comments RFC 8641, Sep. 2019. DOI: 10.17487/RFC8641.

[5] F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network Telemetry Streaming Services in SDN-Based Disaggregated Optical Networks," *Journal of Lightwave Technology*, vol. 36, no. 15, pp. 3142–3149, Aug. 2018. DOI: 10.1109/JLT.2018.2795345.

[6] D. Wen, Y. Zhou, X. Li, Y. Shi, K. Huang, and K. B. Letaief, "A Survey on Integrated Sensing, Communication, and Computation," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 5, pp. 3058–3098, Oct. 2025. DOI: 10.1109/COMST.2024.3521498.

[7] A. Rode, M. Farsi, V. Lauinger, M. Karlsson, E. Agrell, L. Schmalen, and C. Häger, "Machine learning opportunities for integrated polarization sensing and communication in optical fibers," *Optical Fiber Technology*, vol. 90, p. 104047, Mar. 2025. DOI: 10.1016/j.yofte.2024.104047.

[8] K. Kaeval, H. J. Kerm, T. Jarvelill, C. Natalino, and J. Muller, "Vendor Neutrality Drivers and Hindrances - Optical Spectrum as a Service in Disaggregated and Open Networks," in *2025 European Conference on Optical Communications (ECOC)*, Sep. 2025, pp. 1–4. DOI: 10.1109/ECOC66593.2025.11263188.

[9] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection," *Journal of Network and Computer Applications*, vol. 116, pp. 9–23, Aug. 2018. DOI: 10.1016/j.jnca.2018.05.004.

[10] R. Wilton, "YANG-Push Operational Data Observability Enhancements," Internet Engineering Task Force, Internet Draft draft-wilton-netconf-yp-observability-00, Oct. 2024.

[11] R. Casellas, R. Martinez, R. Vilalta, R. Munoz, A. Gonzalez-Muniz, O. G. de Dios, and J.-P. Fernandez-Palacios, "Advances in SDN control and telemetry for beyond 100G disaggregated optical networks [Invited]," *Journal of Optical Communications and Networking*, vol. 14, no. 6, pp. C23–C37, Jun. 2022. DOI: 10.1364/JOCN.451516.

[12] K. Odoh, "An Architecture for Privacy-Preserving Telemetry Scheme," Jul. 2025. DOI: 10.48550/arXiv.2507.06350.

[13] Z. Liu, J. Bi, Y. Zhou, Y. Wang, and Y. Lin, "NetVision: Towards Network Telemetry as a Service," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, Sep. 2018, pp. 247–248. DOI: 10.1109/ICNP.2018.00036.

[14] T. Graf and A. Elhassany, "An Architecture for YANG-Push to Message Broker Integration," Internet Engineering Task Force, Internet Draft draft-ietf-nmop-yang-message-broker-integration-10, Jan. 2026.

[15] Y. Zhou, J. Li, G. Stringhini, A. K. Coskun, and Z. Liu, "Enabling Privacy-preserving Multidimensional Network Telemetry with Autoencoders," in *2023 IEEE International Conference on Cloud Engineering (IC2E)*, Sep. 2023, pp. 228–229. DOI: 10.1109/IC2E59103.2023.00036.

[16] M. Kablan, A. Alsudais, E. Keller, and F. Le, "Stateless network functions: Breaking the tight coupling of state and processing," in *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'17. USA: USENIX Association, Mar. 2017, pp. 97–112.

[17] H. Zhang, Z. Cai, and Y. Li, "An overview of software-defined network measurement technologies," *SCIENTIA SINICA Informationis*, vol. 48, no. 3, pp. 293–314, Mar. 2018. DOI: 10.1360/N112017-00203.

[18] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band Network Telemetry via Programmable Dataplanes," in *ACM SIGCOMM*, 2015.

[19] L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, "In-band Network Telemetry: A Survey," *Computer Networks*, vol. 186, p. 107763, Feb. 2021. DOI: 10.1016/j.comnet.2020.107763.

[20] Y. Kim, D. Suh, and S. Pack, "Selective In-band Network Telemetry for Overhead Reduction," in *IEEE International Conference on Cloud Networking (CloudNet)*, Oct. 2018, pp. 1–3. DOI: 10.1109/CloudNet.2018.8549351.

[21] E. Voit, A. Clemm, A. G. Prieto, E. Nilsen-Nygaard, and A. Tripathy, "Subscription to YANG Notifications," Internet Engineering Task Force, Request for Comments RFC 8639, Sep. 2019. DOI: 10.17487/RFC8639.

[22] A. Sgambelluri, A. Pacini, F. Paolucci, P. Castoldi, and L. Valcarenghi, "Reliable and scalable Kafka-based framework for optical network telemetry," *Journal of Optical Communications and Networking*, vol. 13, no. 10, pp. E42–E52, Oct. 2021. DOI: 10.1364/JOCN.424639.

[23] M. Fedor, M. L. Schoffstall, J. R. Davin, and J. D. Case, "Simple Network Management Protocol (SNMP)," Internet Engineering Task Force, Request for Comments RFC 1157, May 1990. DOI: 10.17487/RFC1157.

[24] E. Gamess and S. Hernandez, "Performance Evaluation of SNMPv1/2c/3 using Different Security Models on Raspberry Pi," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 12, no. 11, Nov. 2021. DOI: 10.14569/IJACSA.2021.0121101.

[25] B. Shariati, H. Qarawlus, S. Biehs, J.-J. Pedreno-Manresa, P. Safari, M. Balanici, A. Bouchedoub, H. Haße, A. Autenrieth, J. K. Fischer, and R. Freund, "Telemetry Framework with Data Sovereignty Features," in *2023 Optical Fiber Communications Conference and Exhibition (OFC)*, Mar. 2023, p. M3G.2. DOI: 10.1364/OFC.2023.M3G.2.

[26] A. Jafari, B. Shariati, A. Mitrovska, P. Safari, M. Balanici, and J. K. Fischer, "NOBS: A Data-Sovereign Telemetry and Monitoring Framework for 6G Networks," in *2025 25th Anniversary International Conference on Transparent Optical Networks (ICTON)*, Jul. 2025, pp. 1–5. DOI: 10.1109/ICTON67126.2025.11125124.

[27] H. Zafar, U. Fattore, F. Cirillo, and C. J. Bernardos, "Data Usage Control for Privacy-Enhanced Network Analytics in Private 5G Networks," *IEEE Open Journal of the Communications Society*, vol. 6, pp. 2976–2992, 2025. DOI: 10.1109/OJCOMS.2024.3522379.

[28] S. Woo, J. Sherry, S. Han, S. Moon, S. Ratnasamy, and S. Shenker, "Elastic scaling of stateful network functions," in *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'18. USA: USENIX Association, Apr. 2018, pp. 299–312.

[29] D. Basin, M. Gras, S. Krstić, and J. Schneider, "Scalable Online Monitoring of Distributed Systems," in *Runtime Verification*, J. Deshmukh and D. Ničković, Eds. Cham: Springer International Publishing, 2020, pp. 197–220. DOI: 10.1007/978-3-030-60508-7_11.

[30] T. P. Raptis and A. Passarella, "A Survey on Networked Data Streaming With Apache Kafka," *IEEE Access*, vol. 11, pp. 85 333–85 350, 2023. DOI: 10.1109/ACCESS.2023.3303810.

[31] L. Velasco, P. Gonzalez, and M. Ruiz, "Distributed intelligence for pervasive optical network telemetry," *Journal of Optical Communications and Networking*, vol. 15, no. 9, pp. 676–686, Sep. 2023. DOI: 10.1364/JOCN.493347.

[32] T. Graf, W. Du, P. Francois, and A. H. Feng, "A Framework for a Network Anomaly Detection Architecture," Internet Engineering Task Force, Internet Draft draft-ietf-nmop-network-anomaly-architecture-07, Jan. 2026.