

# Rethinking the In-band Network Telemetry: Towards Application and Server-Level Network Telemetry

Francisco Germano Vogt  
Universidade Estadual de Campinas (Unicamp)  
Campinas, Brazil

Fabricio Rodriguez  
Universidade Estadual de Campinas (Unicamp)  
Campinas, Brazil

Marcelo Caggiani Luizelli  
Federal University of Pampa (Unipampa)  
Alegrete, Brazil

Christian Esteve Rothenberg  
Universidade Estadual de Campinas (Unicamp)  
Campinas, Brazil

## Abstract

Fine-grained network telemetry is the basis for monitoring modern data centers and supporting in-network applications (INAs). INAs have recently adopted a new telemetry approach in which servers report data for INAs' use. Despite its potential, this method remains complex and unstandardized. This work introduces **SATS**, the **Server and Application Telemetry System**. SATS is the first effort to standardize application and server-level telemetry, offering high abstraction and easy deployment. SATS is a lightweight framework designed to enhance existing INAs' performance and encourage new ones' development.

### ACM Reference Format:

Francisco Germano Vogt, Fabricio Rodriguez, Marcelo Caggiani Luizelli, and Christian Esteve Rothenberg. 2024. Rethinking the In-band Network Telemetry: Towards Application and Server-Level Network Telemetry. In *Proceedings of the CoNEXT Student Workshop 2024 (CoNEXT-SW '24)*, December 9–12, 2024, Los Angeles, CA, USA. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3694812.3699922>

## 1 Introduction

In-network computing (INC) has emerged as a new paradigm in modern networking, leveraging the programmable capabilities provided by SmartNICs and programmable switches to execute computational tasks directly within the network. This paradigm shift offers many benefits, including higher throughput, reduced latency, and enhanced power efficiency. The INC impacts extend beyond traditional network functions such as load balancing and congestion control, allowing the in-network implementation of application tasks like key-value stores and machine learning aggregation.

In-network applications (INAs) are commonly based on network-level metrics like packet counters, inter-packet gaps, and queue lengths. These metrics are monitored and shared using frameworks like in-band network telemetry (INT), which allows switches to produce millions of information reports per second [1]. However, network-level information alone is not always sufficient to operate INAs. Recently, efforts demonstrated that combining network-level information with the application and server-level information can

result in more efficient INAs for use cases like load balancing [2] and task scheduling [3]. These solutions combine network metrics with server metrics like CPU utilization and application load and result in better solutions when compared to the ones based solely on network-level metrics.

Despite existing efforts towards this direction, collecting and using server and application-level information in INAs is not trivial. Reversing the telemetry process by collecting data from servers/apps imposes different constraints compared to traditional INT collection, which cannot be used due to factors such as: (i) *data should be collected on servers and processed by network devices, but the INT specification does not support this inversion*; (ii) *the INT architecture needs to be restructured, as current roles are inadequate; servers and applications must now be incorporated as key elements*.

Since we cannot use well-known frameworks like INT to invert the telemetry process, collecting application and server-level information poses the following challenges:

**How to collect the information?** There is no framework or standardization to collect server and application-level information. This implies that all INAs that use or plan to use this type of information must implement their own strategy to collect and transport information from servers to the INAs.

**How to administrate information collection?** Due to the lack of standardization in data collection, there are no strategies to administrate this collection. Questions such as which data can we collect? How often? How to administrate the collection to avoid repeated data? Has not yet been addressed.

**What is overhead and how to deal with it?** Just as collecting data with INT includes network and processing overhead, collecting data from servers and applications also does – and can be even higher. In INT collection, data is collected from switches capable of processing Tbps of traffic and sent to collectors specialized in processing this information. When collecting data from servers and applications, we include an overhead precisely where the applications are executed, which can impact their performance.

To address these challenges, we propose **SATS**: the **Server and Application Telemetry System**. SATS is a novel telemetry framework designed to collect and manage application and server-level information efficiently. SATS provides a high-level abstraction to gather information from servers to network devices. With SATS fully implemented, we aim to deliver the following contributions: **SATS Standardization**. We seek to provide a complete standardization for SATS collection. This standardization will allow developers to create new INA easily based on SATS. Standardization will also

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

CoNEXT-SW '24, December 9–12, 2024, Los Angeles, CA, USA

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1255-5/24/12

<https://doi.org/10.1145/3694812.3699922>

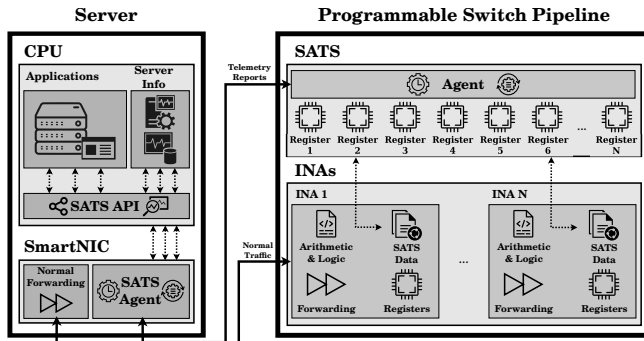


Figure 1: SATS preliminary architecture

help with interoperability between new and existing INAs and will help the integration with other monitoring systems.

**Boosted INA performance.** We will demonstrate how SATS can enhance the performance of current INAs by providing access to a new set of finely granulated information. Our target use cases include INAs for machine learning, load balancing, task scheduling, and traffic classification.

**Novel INA use cases.** We hope that with the standardization and abstraction provided by SATS, new INAs may emerge, working with use cases that have still been little explored in INC. Some of the envisioned SATS-augmented use cases include in-network VNF resource scaling and in-network end-to-end Remote direct memory access (RDMA) routing and congestion control mechanisms.

## 2 SATS Architecture & Design Principles

Figure 1 presents the SATS' preliminary architecture. For simplicity, we consider only one switch and one server. Each switch contains its own set of INAs (e.g., load balancing, heavy hitter detection, congestion control), which contains their control, arithmetic, forwarding logic, and memory. Additionally, each INA may contain SATS data corresponding to server and/or application data collected via SATS telemetry. The SATS' telemetry operates through two core components: the SATS' Agent and the SATS' API, described below:

**SATS Agent.** Operates on both switches and servers, implementing all the telemetry administration logic. The agent is responsible for requesting/receiving SATS reports and controlling factors such as the correct slots to read and store data and the report's frequency. It can be implemented on switches using Programming Protocol-independent Packet Processors (P4), interacting with INAs to provide the requested data. On servers, it can be deployed on SmartNICs, enabling RDMA while avoiding CPU overhead.

**SATS API.** Operates on the server's CPU and is responsible for interacting with applications and system information. Applications should provide the information to the API, which will interact with the agent to send the data.

SATS is designed to allow the switch agent to either passively receive data or actively request it. While registers must be pre-defined at compile time, we expect that the register-to-INA allocation and parameters, like report frequency, can be adjusted at runtime. SATS is also designed to provide two main benefits: (i) *easy integration into servers and INAs*, and (ii) *minimize telemetry overhead*. To achieve these goals, SATS is built on the following design principles: **Lightweight Agent.** The agent is designed to be lightweight on both the switch and server side. This allows most of the switch's

resources to be used to implement INAs and minimize the overhead on the server, not compromising the application's performance.

**High Abstraction Level.** SATS offers a high-level interface for configuring data collection. On the switch side, SATS manages all information collection and only exposes the registers with the collected data to the INAs. On the server side, SATS provides an API communicating with applications to receive the required information. From the API, SATS is responsible for administrating and delivering this information to the destination.

## 3 Discussion

The design of SATS presents open challenges requiring further exploration. Below, we highlight some key issues:

**Routing.** Reversing telemetry involves sending data from servers/app to switches via RDMA requests. A primary challenge lies in effectively addressing programmable switches. Potential solutions include implementing a minimal TCP/IP stack on the switch (e.g., to handle ARP responses) or using spoofed IP/MAC addresses to enable routing in IP networks.

**Network Visibility.** To ensure accurate delivery of telemetry reports, the SATS agent on the server must be aware of the network switches intended to receive telemetry data. Switches could request telemetry by broadcasting messages, or alternatively, a control plane application could inform the servers about relevant switches.




**SATS Agent.** The SATS' agent must manage which telemetry data to send to each switch in the network. Switches may request specific telemetry features, and the agent should determine the appropriate timing and frequency for sending this data.

**Security.** SATS assumes that multiple applications can operate concurrently on the same switch, e.g., via a Tofino hypervisor. However, this raises security concerns, as co-located applications might gain unauthorized access to registers. A possible solution is extending the hypervisor to enforce isolation between applications.

## 4 Closing Remarks

This work introduced SATS, the first telemetry system to collect application and server-level information. We designed SATS as a lightweight framework for easy integration with servers and INAs. While many issues remain in SATS's design and implementation, we plan to address them based on community needs and feedback.

## Acknowledgments

Work supported by Ericsson Telecomunicações Ltda. , and by the Sao Paulo Research Foundation , grant 2021/00199 – 8, CPE SMARTNESS . Also, this work was partially supported by FAPESP grants 2023/00794-9 and 2021/06981-0, and by FAPERGS grant 24/2551-0001394-6. Finally, this study was partially funded by CAPES, Brazil - Finance Code 001.

## References

- [1] Jonatan Langlet et al. 2023. Direct Telemetry Access. In *Proceedings of the ACM SIGCOMM 2023 Conference*. 832–849.
- [2] Hesam Tajbakhsh et al. 2024. P4Hauler: An Accelerator-Aware In-Network Load Balancer for Applications Performance Boosting. *IEEE Transactions on Cloud Computing* (2024).
- [3] Parham Yassini et al. 2024. Horus: Granular {In-Network} Task Scheduler for Cloud Datacenters. In *21st USENIX NSDI* 24. 1–22.