Deep dive into OpenTelemetry for evaluation of their observability in edge computing environment

Omar Bin Kasim Bhuian
Computer Science Department,
Huaiyin Institute of Technology,
Huai'an, Jiangsu, China
omarbinkasimsefat@gmail.com

Ki-Woong Park*
Department of Information Security and Convergence
Engineering for Intelligent Drone,
Sejong University,
Seoul 05006, South Korea
woongbak@sejong.ac.kr

Abstract
The rise of cloud-native infrastructures has accelerated the adoption of OpenTelemetry as the leading standard for monitoring system performance. OpenTelemetry's well-established utility in software observability gives way to significant limitations when applied to hardware and real-time systems. This paper explores the challenges of integrating OpenTelemetry with hybrid cloud-hardware environments, drawing on real-world use cases such as the Israel-Lebanon smart device sabotage. By highlighting performance deficiencies, protocol incompatibilities, and security vulnerabilities, we illustrate how OpenTelemetry's design shortcomings diminish its utility in hardware-intensive applications. We offer recommendations for improving device compatibility and mitigating cybersecurity threats. The document seeks to provide an exhaustive analysis and a strategy for enhancing telemetry in hybrid settings.

Keywords: open-telemetry, cloud-native infrastructure, distributed systems, hardware monitoring, security Vulnerabilities, IoT.

I. Introduction

A. Background

In the present day, the scenario is even more intricate with apps and intelligent things coming in play with embedded hardware systems. To manage system monitoring and analysis in such conditions, OpenTelemetry is emerging as an open-source solution for observability with the ability to capture distributed traces, metrics, and logs. The strength of this tool is that deployment environments where microservices are critical and you need powerful monitoring, debugging tools for high availability and well-performance support [1].

As we see cloud infrastructures include hardware-centric environments such as devices tackling new workloads, the shortfalls of OpenTelemetry also become more pronounced. IoT systems are normally based on low-power processors. Not have expansive memory, which is best for traditional telemetry tools as they require a lot of resources. Or there are also the issues related with hardware settings, as OpenTelemetry's native design can be challenging due to the multiple communication protocols used on hardware [2], [3].These challenges, in turn, contain threats for time-critical safety-ensuring systems. Open telemetry is crucial in deter catastrophic failures scenarios like smart city infrastructures or industrial control systems.

B. Problem Statement

This post digs into the OpenTelemetry technology shortcomings by focusing on its bloated resource usage in hybrid cloud and hardware environments. These systems are increasingly being integrated with the Internet of Things (IoT) devices, for which novel solutions have to be developed given their constrained computing power and communication protocols. In this paper, the authors take the Israel-Lebanon conflict as an example of applying OpenTelemetry and look at real world examples to describe situations where it may be challenging to use OpenTelemetry, especially in scenarios that require lightweight use of resources and real time monitoring for system protection [2], [4]. We will discuss methods to enhance OpenTelemetry's infrastructure, including the incorporation of additional protocols and improved resource utilization, to provide more comprehensive telemetry in mixed situations.

II. OpenTelemetry in Cloud-Native Environments

A. Cloud-Native Use Cases and Benefits

OpenTelemetry has demonstrated considerable advantages in cloud-native systems characterized by microservice architectures. A case study on financial systems demonstrated how OpenTelemetry facilitates the identification of performance bottlenecks by matching telemetry data across many microservices. In this context, OpenTelemetry's capability to provide distributed tracing is crucial for resolving issues related to complex, asynchronous service interactions [1], [3].

In cloud-native systems, telemetry collecting, analysis, and visualization tools, such as Prometheus and Grafana, are

crucial for managing high traffic and system scalability. OpenTelemetry's integration with Prometheus, for example, enables deep visibility into Kubernetes clusters, offering multi-dimensional data insights that support real-time decision-making [5]. OpenTelemetry's resource overhead is mostly due to the volume of telemetry data and the demands of real-time processing. To fix this, adaptive sampling and lightweight agents can improve observability while putting as little stress as possible on devices with limited resources. As system complexity escalates, the resource overhead of OpenTelemetry becomes evident. Implementing OpenTelemetry in cloud infrastructure led to a 46.5% rise in CPU utilization and a 47.5% increase in memory usage, raising scalability issues in bigger settings [3]. Case studies, including its application in financial systems, demonstrate the advantages of OpenTelemetry in cloud-native environments, especially in performance monitoring and issue diagnosis within microservice architectures. These observations can guide future research focused on creating lightweight telemetry solutions tailored for IoT environments, where efficiency and minimal resource use are essential.

Table 1: Performance overhead encountered in cloud-native deployments using different models of OpenTelemetry [3].

Deployment Model	**CPU Overhead**	**Memory Overhead**
Daemonset	46.5%	47.5%
Sidecar	50.0%	52.0%

B. Case Study1: Israel-Lebanon Conflict and IoT Vulnerabilities

The security flaws in IoT systems, especially those dealing with vital infrastructure, were exposed during the Israel-Lebanon conflict. There have been rumors that Israel compromised Lebanese infrastructure using cyber warfare tactics, namely targeting smart equipment such as electrical appliances that may be manipulated remotely and various Internet of Things devices. Reports indicate that these devices were utilized as hacking weapons by the Israeli military, leading to physical harm and the remote detonation of specific smart devices [6]. The incident underscores the growing risks associated with adding IoT technologies to vital systems. Absence of strong telemetry systems meant that the hardware components, including IoT devices, were not fully supervised and hence it left a window open to potential attacks. OpenTelemetry works very well for cloud-native software, but in this case would not be good enough as it does not provide a way to send hardware telemetry.
IoT devices often have resource constraints, such as limited power, low computational capacity and multiple communication protocols. That means the OpenTelemetry architecture, which is mainly focused on cloud-native microservices, doesn't fit very well IoT devices. In high-risk scenarios such as this, the lack of real-time monitoring and ability to identify anomalies instead of devices (in IoT) diminishes the framework [1], [2].

C. Case Study 2: Mirai Botnet Attack on IoT Devices

The 2016 Mirai botnet assault exemplifies significant IoT vulnerabilities inside hybrid cloud hardware ecosystems. The Mirai botnet used inadequate security configurations to target several IoT devices, including IP cameras, routers, and DVRs. These machines launched a massive DDoS attack after being compromised, affecting global internet and cloud infrastructures [7]. Lack of real-time monitoring and telemetry solutions for IoT devices helped the attack succeed. The majority of impacted devices' designs inhibited their ability to transmit telemetry data that could have notified administrators of their breach. Because OpenTelemetry is focused on the cloud, it doesn't yet have the right tools to connect with IoT devices easily for real-time monitoring. This coverage gap let the attack spread quickly. If real-time telemetry had been available across all of the infected IoT devices, strange behavior could have been caught early on, like when the devices joined a botnet and sent out strange traffic. This could have stopped the attack before it got too big [8].

III. Deep Dive into the Technical Failures:

OpenTelemetry does not have native support for the protocols used by many IoT devices in critical infrastructure, like Zigbee, MQTT, and Bluetooth. Its capacity to provide complete end-to-end observability is compromised by this constraint. The cyberattack during the Israel-Lebanon conflict exploited weaknesses by targeting unsecured IoT devices, underscoring the urgent requirement for telemetry systems that provide real-time hardware monitoring. OpenTelemetry's focus on software and cloud environments makes it much less useful for finding security holes or performance problems in hardware environments with limited resources [8].

The Mirai botnet assault exploited inherent IoT security vulnerabilities, such as default passwords and obsolete firmware. Nonetheless, a more significant concern was the lack of real-time telemetry to identify and counteract the attack promptly. The scale and variety of IoT protocols revealed OpenTelemetry's shortcomings in hardware telemetry. A multitude of IoT devices are inadequate. OpenTelemetry imposes overhead for logging and trace collection, which many IoT devices are unable to handle, hence intensifying the difficulties of protecting these environments [5].

A. Protocol Incompatibility and Integration Issues

Although OpenTelemetry is intended for cloud-native environments, its ideas can be modified for hardware contexts. It is crucial to recognize its limitations, but equally critical to accept the framework's merits in delivering

comprehensive observability. By utilizing its current features and suggesting specific improvements, we can develop a more adaptable tool for hybrid systems. Telemetry systems are essential for the effective monitoring and management of modern infrastructures, including IoT and hardware-intensive environments. OpenTelemetry, specifically designed for cloud-native applications, is a powerful solution for augmenting telemetry capabilities across multiple domains. However, its current limitations, especially the lack of native support for hardware-specific communication protocols like Zigbee and MQTT, diminish its efficiency in resource-constrained applications. Real-time telemetry is essential in hazardous environments, such as disaster management and industrial IoT. This underscores the necessity to modify OpenTelemetry's framework to address these challenges [1], [2].

Furthermore, hardware systems impose stringent resource constraints, rendering the significant overhead of OpenTelemetry inappropriate for low-power devices. Real-time telemetry systems, particularly those employed in disaster management, can be significantly compromised by even a minor delay in data collection. Real-time systems in flood-prone areas rely on instantaneous data to predict water levels and issue alerts. Inefficient telemetry systems can lead to catastrophic failures caused by delays [4], [9].

Figure 1 shows how Open Telemetry maximizes data flow in cloud-native systems, mostly using HTTP and gRPC protocols. Hardware systems, including IoT devices, however, rely on multiple protocols such as Zigbee, Modbus, and MQTT, which OpenTelemetry does not naturally support. The differences in hybrid infrastructures preclude thorough monitoring, which makes important devices vulnerable to unnoticed failures and security flaws. Improving protocol support will enable flawless telemetry collection in hardware as well as cloud-native environments, hence addressing these problems [3], [8].

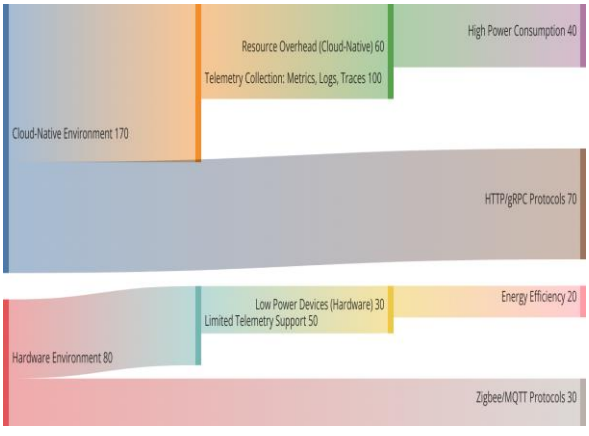

Figure 1: A conceptual graphic that highlights the protocol and architectural contrasts between OpenTelemetry's limited capabilities in hardware contexts and its data flow in cloud-native systems.

B. *Resource Constraints and Performance Overhead*

The resource limits in IoT and embedded systems intensify the shortcomings of OpenTelemetry. The additional CPU and memory burden from telemetry gathering is significant in cloud-native systems; however, in hardware contexts, it may result in substantial performance deterioration. Cheikhrouhou et al. [10] studied disaster management telemetry systems and found that adding real-time monitoring tools caused a latency increase of more than 10%, which meant that people took longer to react in critical emergency situations.

Table 2: Impact of telemetry collection on real-time systems [10].

Real-Time System	**Latency Increase with Telemetry**	**Impact**
Flood Warning	+10%	Delayed alerts
IoT Smart Grid	+8%	Reduced response time

IV. Enhancing OpenTelemetry for Hardware Monitoring

A. *Protocol Extension and Lightweight Agents*

For OpenTelemetry to become a real contender in hardware environments, it will need to be improved. As such, OpenTelemetry should start by enabling these hardware-specific protocols, such as Zigbee and MQTT, which are ubiquitous in IoT environments through modular protocol extensions. Potentially, using lightweight telemetry agents for low-power devices might reduce these costs of resources where OpenTelemetry is being used. Lots and lots of efficient trace, metrics, and log collection going on here—all in a manner that the agent can perform real-time monitoring without excessive pressure on the device's computational resources. This offers a path to continue using OpenTelemetry in environments that are hardware and resource-intensive as well as resource-constrained, helping bridge the gap between cloud-native and IoT systems [2]. In addition, a small resource overhead is required: in embedded systems, a lightweight telemetry agent that is optimized for low-power devices should be used. Those agents need to look at key metrics that can reduce the number of calls required to have real-time data processing running without requiring much CPU and I/O from the system.

The further development of dynamically loadable/unloadable telemetry collectors in OpenTelemetry (modularized) may also strengthen the same flexibility for hybrid environments [10].

B. *Security and Privacy Enhancements*

The story of Israel-Lebanon demonstrates a strong need for system monitoring and telemetry around critical infrastructure To keep their telemetry data for the hardware

systems tight and access-controlled to avoid being exploited for harmful reasons [1], [4]. OpenTelemetry will have to as well end-to-end encryption and robust mechanisms to assure that telemetry does not become a trojan horse for cyberattacks, especially in privacy-sensitive application surroundings like smart cities or industrial IoT.

V. Conclusion:

There is no doubt that Open Telemetry is a powerful tool for cloud-native observability but the points of improvement in hardware-centric systems must be taken care before it becomes an end-to-end solution for infrastructure, as we see today. Enhancing Security, Supporting Lightweight Device, and More Protocol This way, Open Telemetry will be a strong and flexible framework. This work demonstrates from technical analysis and case studies that it is not only possible but imperative to close the chasm between cloud-native telemetry and hardware-derived telemetry for hybrid system observability.

ACKNOWLEDGEMENT

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (Project No. RS-2024-00438551, 50%; 2022-11220701, 30%), and the National Research Foundation of Korea (NRF) grant funded by the Ministry of Science and ICT (Project No. RS-2023-00208460, 20%).

References

[1] D. Gomez Blanco, *Practical OpenTelemetry: Adopting Open Observability Standards Across Your Organization*. Berkeley, CA: Apress, 2023. doi: 10.1007/978-1-4842-9075-0.

[2] O. V. Talaver and T. A. Vakaliuk, “Telemetry to solve dynamic analysis of a distributed system,” *J. Edge Comput.*, vol. 3, no. 1, pp. 87–109, May 2024, doi: 10.55056/jec.728.

[3] E. Norgren, “OPTIMIZING DISTRIBUTED TRACING OVERHEAD IN A CLOUD ENVIRONMENT WITH OPENTELEMETRY”.

[4] G. Suciu, C. Istrate, D. Filip, A. Scheianu, and M. Cigale, “Real-Time Telemetry System for Emergency Situations using SWITCH”.

[5] W. Pourmajidi, L. Zhang, J. Steinbacher, T. Erwin, and A. Miranskyy, “A Reference Architecture for Observability and Compliance of Cloud Native Applications,” Feb. 22, 2023, *arXiv*: arXiv:2302.11617. Accessed: Sep. 23, 2024. [Online]. Available: http://arxiv.org/abs/2302.11617

[6] “EXPLAINED | Cyber Attack on Hezbollah: Pagers Explode, Killing 9 in Lebanon - Frontline.” Accessed: Sep. 28, 2024. [Online]. Available: https://frontline.thehindu.com/news/lebanon-hezbollah-cyber-attack-pager-explosions-warfare-israel-gaza/article68654302.ece

[7] M. Antonakakis *et al.*, “Understanding the Mirai Botnet”.

[8] B. B. Joanna Kosińska and M. M. Marek Konieczny, “(PDF) Toward the Observability of Cloud-Native Applications: The Overview of the State-of-the-Art,” *ResearchGate*, doi: 10.1109/ACCESS.2023.3281860.

[9] “IoT Market View, 2020-2021,” Gartner. Accessed: Sep. 23, 2024. [Online]. Available: https://www.gartner.com/en/documents/4010081

[10] O. Cheikhrouhou, A. Koubaa, and A. Zarrad, “A Cloud Based Disaster Management System,” *J. Sens. Actuator Netw.*, vol. 9, no. 1, Art. no. 1, Mar. 2020, doi: 10.3390/jsan9010006.