Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080115

REVIEW

Monitoring and Observability in Edge Computing Systems: Taxonomy, Comparative Analysis, and Research Directions

Hamza Ahmed[1], **Hassan Jamil Syed**[2,*], **Aqsa Aslam**[1], **Sehar Zehra**[1], **Ummay Faseeha**[1] and **Nurzati Iwani Othman**[2]

[1]School of Computing, National University of Computer and Emerging Sciences, Karachi, Pakistan
[2]School of Technology, Asia Pacific University of Technology & Innovation, Kuala Lumpur, Malaysia

*Corresponding Author: Hassan Jamil Syed. Email: hassan.jamil@apu.edu.my

Received: 03 February 2026; Accepted: 30 April 2026; Published: 15 June 2026

ABSTRACT: Edge computing is an emerging model for latency-sensitive and distributed applications. However, the observability of edge computing systems in heterogeneous environments remains a challenge, as most existing approaches are limited to only the system, service, application, and network layers. This paper surveys state-of-the-art solutions for edge observability and monitoring. The paper further introduces a thematic taxonomy that groups the state-of-the-art edge observability and monitoring literature based on monitoring intent, telemetry indicators, observability scope, architectural layers, deployment environments, and observability toolchains. Finally, we compare representative solutions in terms of latency, system overhead, bandwidth consumption, and detection accuracy. Our analysis reveals the trade-offs present in existing solutions regarding the overhead and performance metrics, as well as the limitations they impose when scaling, addressing mobility, or augmenting telemetry. Lastly, we conclude with a discussion on open challenges and future directions for designing next-generation, edge observability frameworks.

KEYWORDS: Edge computing; edge monitoring; monitoring solutions; observability; telemetry; microservices; performance analysis; distributed systems

1 Introduction

In recent years, there has been an increasing shift in computing towards edge computing. This is achieved by shifting information and data processing closer to the point of origin, which was previously done at centralized servers. This computing shift offers several advantages in terms of lower latency, bandwidth optimization, and enhanced data security [1,2]. It can be done by the reduction in physical distance between devices and servers. Edge computing has become essential for applications where the latency needs to be minimal, even a millisecond matters, such as autonomous vehicles and industrial robotics [3–5]. The transmission delay is reduced by processing the data locally, which reduces the latency and requires less bandwidth. It preserves bandwidth usage and ensures more reliable data transfers. However, with these benefits, there needs to be a look at security and management issues that may arise with these solutions [6,7].

For different cases, there are different edge architectures available. The hierarchical models process data layer by layer across the architecture, moving from local devices up to regional servers or the cloud for heavy lifting. This structure provides a high degree of control and central oversight [8,9]. In contrast, flat architectures favor direct peer-to-peer communication. For a low number of devices, this simplifies the setup, whereas the management becomes increasingly difficult as the network grows. Fog computing has emerged

as a middle layer to balance the load between the edge and the cloud as a bridge, to maintain low latency while providing extra storage and compute power [2,10–12].

In the field of transportation, healthcare, and smart cities [2,13,14], edge computing has a direct influence on manufacturing. In healthcare, real-time patient monitoring is done by edge computing using data analysis through wearable devices and remote sensors, which helps in improving patient care and outcomes [2,15]. The use of connected vehicles, as well as intelligent traffic management systems in the transportation industry, requires low-latency processing of data to ensure the safety and efficiency of these systems. Smart cites use edge computing to allocate and monitor resources, improve safety, and control pollution [13,15].

Fig. 1 illustrates the conceptual role of eBPF (Extended Berkeley Packet Filter) in enhancing edge monitoring across multiple operational domains. It enables efficient kernel-level observability, and eBPF strengthens the monitoring and performance capabilities that are essential in modern edge computing applications. Edge computing itself plays an essential role in the sectors of manufacturing, healthcare, transportation, and smart cities, where the decision-making needs to be done quickly and critically. In healthcare, real-time monitoring is done through wearable sensors and medical IoT devices that depend on low-latency processing at the edge to have continuous monitoring of patient assessment and early anomaly detection. Similarly, transportation systems that include connected vehicles and intelligent traffic management depend on fast processing to maintain safety and operating efficiency. Smart city infrastructures benefit from edge-enabled monitoring for resource optimization, environmental sensing, and public safety management.

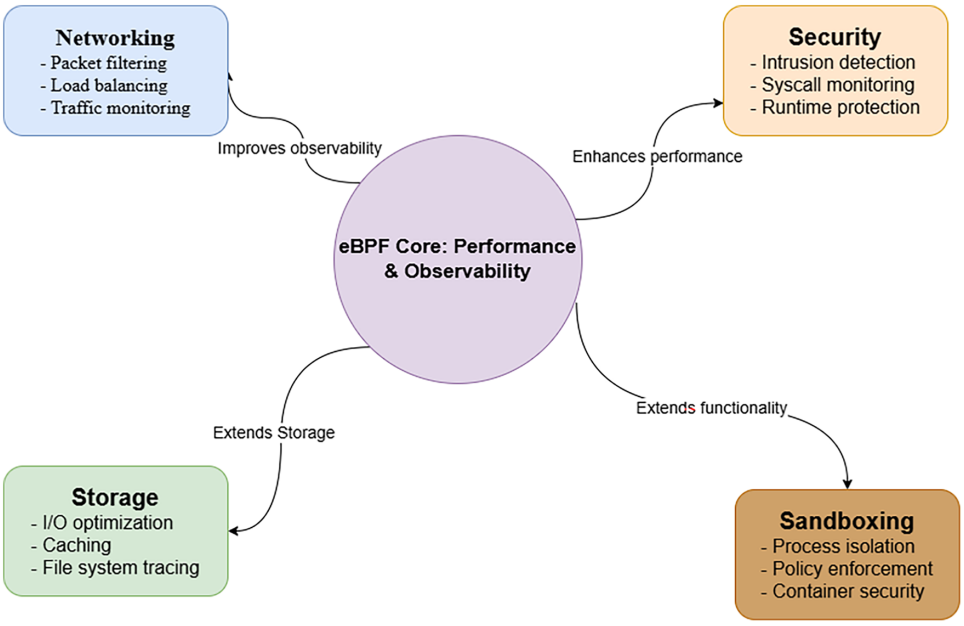

Figure 1: eBPF-driven monitoring architecture for edge environments.

In Fig. 1, the use cases of eBPF highlight how it improves observability and performance. This includes network tracing, runtime security enforcement, storage I/O optimization, and container-level isolation. In the distributed edge environment, the eBPF provides the mechanisms that are required for reliable real-time monitoring. Therefore, edge computing is adapted very highly due to its observability capabilities enabled by eBPF, which allows these systems to address complex, real-world challenges across diverse industrial and social domains.

In the future, it is clearly seen that edge computing is closely connected with emerging technologies, which include 5G networks [16–19], the Internet of Things (IoT) [20,21], and artificial intelligence (AI) [3]. The integration of 5G networks is ideal for real-time edge networks and applications that promise faster data speeds and lower latency. The IoT devices' data can be processed locally at edge servers because it generates large amounts of data, which reduces the need for constant cloud communication. Furthermore, deploying the machine learning algorithms and AI applications on edge devices allows real-time data analysis, which helps in decision-making, significantly enhancing the capabilities of edge computing systems [22]. However, security concerns required a widespread adoption will require addressing security concerns, standardizing protocols, and exploring novel applications across various industries.

In this paper, we present a detailed survey of the state-of-the-art edge monitoring solutions. To the best of our knowledge, only a limited number of existing studies have comprehensively reviewed cloud-edge monitoring solutions, which highlights the uniqueness and significance of our work. For instance, previous studies [23–27] have explored various features of edge-cloud monitoring, but have been deficient in considering all relevant dimensions comprehensively. In this survey, we cover a wide range of critical aspects of edge monitoring solutions, including architecture, monitoring perspectives, communication models, scalability, and the overhead of these solutions. These features are essential in defining the attributes and effectiveness of monitoring solutions.

The rest of the paper is as follows. Section 2 reviews related work on edge monitoring solutions. Section 3 proposed the thematic taxonomy for edge monitoring solutions. Section 4 provides an overview of the state-of-the-art edge monitoring solutions. Section 5 discusses open research issues and challenges in edge computing. Section 6 presents a conclusion of the whole paper to present the findings.

2 Related Work

This section reviews and analyses existing studies that are related to edge computing-based monitoring and observability frameworks. The study examines their proposed architectures, evaluation methodologies, and application domains. The discussion highlights the key contributions and limitations of these works, with particular emphasis on performance monitoring, anomaly detection, and reliability analysis in distributed edge environments. Furthermore, the motivations and research gaps identified in the literature are critically examined to establish the need for more lightweight, accurate, and scalable monitoring solutions suitable for resource-constrained edge systems.

2.1 Research Motivation

The fast-evolving mobile ecosystem and the intersections of new service paradigms, comprised of Internet-of-Things (IoT) ecosystems and sensor-based services, are driving the need for more real-time computational capabilities at the edge of the network. With new workloads and the adaptive allocation of resources with very low latency, and the availability of cloud resources in the center of the network, the classical cloud-based architectures are not effective. The growing number of mobile and IoT devices generates constant streams of data, and sending all the data to cloud data centers adds communication delays, soaks the bandwidth of the network, and increases energy consumption. With the described challenges, Mobile Edge Computing (MEC) appears to be one of the most important paradigm shifts for the design of the network, allowing for real-time computation to be done closer to the sources of data.

Task scheduling and offloading, in terms of the advantages, still remain difficult challenges in MEC. These challenges are the result of the irregular nature of the wireless channel's quality, the mobility of users, the various levels of capability of devices, the workload, the utilization of edge devices, and the position of the edge servers [28,29]. Transfer events, where devices transition between network coverage zones, can interrupt

or degrade the performance of a task being executed without the scheduling controlling for the mobility pattern. All of the above-described complexity and interrelated factors contribute to the determination of a suitable task placement. The number of factors describing the system can be used to create a multi-dimensional model, which can be used to describe task placement and latency, the cost of the data to be sent, the energy consumption of the devices, and the computation of the devices [30].

Table 1 provides a consolidated perspective on existing edge computing and monitoring approaches, a structured comparison of representative studies. In this work, we have explored diverse architectural models, monitoring strategies, and edge-enabled applications, all of which have their contributions that vary significantly in terms of empirical validation, scalability, and practical deployment. A comparative assessment of these models will help in identifying both their strengths and limitations, thereby illustrating unresolved challenges in real-time edge monitoring and observability. As shown in Table 1, the primary edge computing studies provide a conceptual and architectural basis focusing on the reduction of the latency and support of the mobility, but the empirical evaluation is missing. Then, studies started emerging that provided domain-specific monitoring solutions, especially in the fields of healthcare, industrial systems, and smart infrastructure, which showcased optimized responsiveness and on-site analytics. Nevertheless, they were still constrained with regard to the application scope, with limited privacy, and minimal scalability. Contemporary kernel-level and smart monitoring solutions provide precise time, low latency, and superior detection accuracy. However, they tend to be (Operating System) OS-dependent and have additional training and deployment complexities. This outlines the gap present in theoretical literature and practical edge monitoring systems that are scalable and applicable. Integrated observability is lacking in the current MEC scheduling models. Observability is the ability to monitor the system's behavior, patterns of performance, the use of resources, and responsiveness. The distributed and heterogeneous nature of MEC systems does make real-time monitoring more difficult than in centralized cloud systems. This lack of observability makes it difficult to monitor the system's performance. From the absence of observability, scheduling algorithms in MEC systems perform poorly. They are simply unable to handle the runtime scenarios, emerging bottlenecks, and provide consistent Quality of Service (QoS).

Table 1: Strengths, and limitations of existing edge monitoring solutions.

Proposed Method/Reference	Year	Tool/Technique Used	Strengths	Limitations
Secure IoT Service Architecture (BOE) [12]	2020	Trust Evaluation + Cloud/Edge Cooperation	Fast convergence; accurate malicious device isolation	IoT-focused; limited generalization to other domains
DeSFAM (eBPF + AI Threat Monitoring Framework) [28]	2024	eBPF + VAE + Isolation Forest	96% detection accuracy; blocks real CVE attacks; minimal overhead	Requires kernel-level access; ML training cost
The Emergence of Edge Computing [31]	2019	Cloudlets, Fog Computing Model	Foundational EC model; reduces latency; improves mobility support	Conceptual only; lacks empirical evaluation

(Continued)

Table 1 (continued)

Proposed Method/Reference	Year	Tool/Technique Used	Strengths	Limitations
Edge Computing: Vision and Challenges [32]	2017	Edge–Cloud Hierarchy, Service Offloading	Identifies full EC challenge landscape; comprehensive vision	No implementation; no monitoring experiments
Determining Edge Node Real-Time Capabilities (eBPF) [33]	2022	eBPF, kprobes, High-precision Timing	Nanosecond precision; extremely low overhead	Linux-only; no multi-node evaluation
Demystifying Performance of eBPF Network Applications [34]	2025	eBPF Benchmarking Suite	Reveals performance bottlenecks; precise evaluation	No security or monitoring pipeline evaluation
Real-Time Monitoring and Analysis (Industrial Edge) [35]	2019	Edge–Cloud Collaborative Sensing	Enhanced industrial stability; real-time diagnosis	Not validated in large-scale distributed factories
Mobility & Dependence-Aware QoS Monitoring (ghBSRM-MEC) [36]	2021	Bayesian Monitoring + KNN	Higher QoS accuracy under mobility; reduces monitoring deviation	Small dataset; moderate computational overhead
Edge-Based Insulator Self-Explosion Monitoring [37]	2021	SSD, MobileNet, Multi-model Fusion (YOLO, RCNN)	High recognition accuracy (~95%); lowers communication cost	Domain-specific; limited scalability beyond grids

To overcome these challenges, there is a need for a comprehensive scheduling framework that is supported by observability and capable of making intelligent task offloading decisions while considering dynamic network behavior and changing performance conditions. The motivation of this research is to develop a cost-aware and mobility-enhanced task scheduling approach that integrates latency modeling, energy consumption estimation, and transmission delay analysis. There are some limitations in the current monitoring solution as well, including scheduling methods, a lack of mobility support, no performance modelling, and poor system visibility. This work is an effort toward the development of more adaptive, efficient, transparent edge computing. The goal of this work is not only to improve theoretical understanding but also the design of next-generation MEC systems, which can provide reliable and real-time computation services for modern mobile, IoT, and distributed applications.

2.2 Research Objective

The objective of conducting this study is to offer a thorough insight into the observability frameworks and tools that have been specially developed for containerized microservices. The primary contribution of this study is the establishment of a thematic taxonomy, which would serve to introduce a sense of order to

a field of study that, to the current state, has not had a systematic means of categorizing edge monitoring tools and applications developed for such purposes in place. In addition to offering a means of systematic categorization, the findings and methodology of this study will offer a comparison that reveals the true effects of such tools on system reliability and performance.

2.3 Data Collection Process

In the process, searches were conducted using different libraries i.e., IEEE Xplore, ACM Digital Library, and Google Scholar, utilizing targeted keywords such as:

- Observability in micro services
- Containerized environments
- Micro services Performance Monitoring in Kubernetes
- Observability in Cloud-Native Applications

Additionally, industry-standard tools such as OpenTelemetry, Prometheus, and Grafana were selected due to their widespread adoption. The collected data informed the thematic taxonomy and facilitated the comparative analysis of observability solutions.

2.4 Selection of Observability Frameworks and Tools

In the selection process, researchers have suggested exploring other ranked frameworks. The technologies involved in these frameworks were capable of providing relevance to micro services and containerized environments. There are some industry standard solutions like Prometheus, Loki, Jaeger available, as well as tools made for cloud computing (or containerized composite apps) to solve new problems. The assessment was centered on three primary observability characteristics:

- Metrics collection
- Distributed tracing
- Log aggregations

These criteria provided comprehensive insights into each framework's role in enabling observability within microservices architectures.

2.5 Inclusion and Exclusion Criteria

Inclusion Criteria

- Studies on task scheduling, computation offloading, or performance optimization in MEC.
- Research offering analytical models for latency, transmission delay, or energy consumption.
- Work addressing mobility-aware decision-making or dynamic wireless conditions in edge systems.
- Studies incorporating observability elements such as performance monitoring or resource tracking.
- Research presenting cost models, optimization formulations, or mathematical frameworks for edge task execution.
- Recent studies comparing local and edge execution performance.

Exclusion Criteria

- Studies limited to cloud computing with no edge involvement.
- Research on mobile performance is lacking in scheduling or offloading aspects.
- Models missing latency, energy, or transmission cost components.
- Papers providing only conceptual architecture without analytical evaluation.

- Outdated work (pre-2017) unless foundational to MEC.
- Solutions designed for monolithic or centralized systems without mobility or distributed execution.

The review and selection process establishes a clear foundation by identifying relevant studies, frameworks, and evaluation criteria within the domain of edge computing and observability. Building upon this foundation, Section 3 introduces a structured taxonomy that organizes these selected works into coherent categories. This transition enables a more analytical perspective, where the diverse approaches identified earlier are thematically classified based on common parameters, thereby facilitating a deeper understanding of patterns, relationships, and gaps in edge monitoring solutions.

3 Taxonomy of Edge Monitoring

This section presents a thematic taxonomy of existing edge monitoring solutions discussed in the literature, with particular emphasis on their classification. The proposed classification is based on a set of parameters that frequently appear across the majority of related works, as shown in Fig. 2. The parameters that were chosen and used in this thematic taxonomy include Monitoring Intent, Telemetry Indicators, Observability Scope, Architectural Layers, Deployment Environment, Observability Toolchain, and Architecture Paradigms. Each dimension incorporates refined subtopics that consistently emerge across modern monitoring frameworks. Monitoring Intent captures goals related to performance optimization, real-time responsiveness, and security and reliability assurance. Telemetry Indicators encompass the key signals used to characterize system behaviour, including latency-related measures, resource consumption indicators, and quality-of-service metrics. Observability Scope distinguishes whether monitoring is focused on device-level behaviour, edge-node execution, or application-level operations. This is where the computing actually happens, in Architectural Layers. The precise computing happens in the architectural layers, be it fog, edge, or cloud. The deployment setting denotes the place of the physical hardware and where monitoring is to be performed, such as well-endowed cloud data centers or lean devices at the edge. For us to reason about and troubleshoot what's going on in these systems, we need our observability toolchain to gather all this data and let us query it. Architectural paradigms specify how such disparate systems are architected and constructed, for example, edge–cloud symbiosis, Edge AI, or the increasing reliance on containerized micro services.

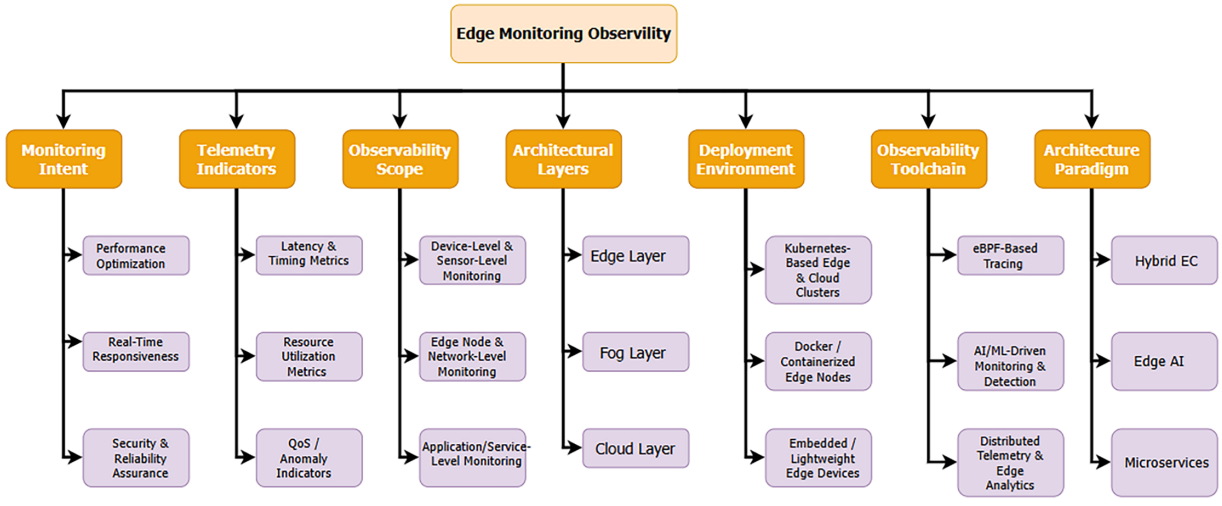

Figure 2: Conceptual comparison of cloud and edge computing architectures.

This taxonomy provides a clear and logical structure that brings together practical methods, such as eBPF-based tracing, on-device inference, and fault stress testing, with theoretical models. By bridging this gap, it helps us identify patterns in a systematic way and maintain a consistent focus, ultimately allowing different monitoring studies to connect and complement each other more effectively.

3.1 Monitoring Intent

As shown in Fig. 2, the monitoring intent is concerned with what should be measured in an edge computing system. In the scope of traditional computing, resources at edge environments are limited and workloads are changeable abruptly and cannot afford delay. If you're new to the concept of monitoring intention, its core is finding a way to safeguard your system and make it always available and secure. To provide such monitoring, it requires monitoring edge nodes in a continuous manner so that the nodes are kept live and viable despite sudden changes in the network.

3.1.1 Performance Optimization

The main concern for performance optimization is to keep things stable and use fewer resources. Since the nodes on the edges have fewerresources in terms of processing capability, a more accurate measurement of CPU use and memory patterns as well as task processing, becomes essential. This will enable more informed workload distribution and model adjustment to avoid the bottlenecks that are characteristic of resource-constrained hardware, even as the system maintains a high level of efficiency due to the changeable workload.

3.1.2 Real-Time Responsiveness

In most edge use cases, having the ability to respond to local occurrences on the spot can be taken as an indicator of success. Real-time monitoring ensures that an immediate feedback cycle is achieved to fulfill the latency requirements of higher intensity by monitoring queue accumulation and event distribution at the time of occurrence. For instance, in applications such as industrial sensing or auto-automation, real-time observation of phenomena ensures that critical decisions, for instance, at the time of observation, are taken at an edge point and not after an entire round-trip to a cloud point.

3.1.3 Security & Reliability Assurance

Security and reliability mainly focus on maintaining the stability, integrity, and reliability of edge systems. The goal of monitoring is to identify anomalies, detect unauthorized or hazardous behavior, and evaluate system strength at both the software and hardware levels. Since edge nodes are deployed in distributed and often unprotected environments, monitoring acts as a safeguard against resource misuse, abnormal patterns, and potential intrusions. Some reliable monitoring systems also detect early signs of degradation, sensor faults, or performance instabilities that could lead to failures. Considering all these together, this ensures continuous, safe, and predictable operation across decentralized edge infrastructures.

3.2 Telemetry Indicators

Fig. 2 highlights the different types of telemetry indicators, focusing on the various ways to measure system behavior in edge environments. This includes the measurement of the various ways in which resources are utilized in the different distributed computing, communication, and resource computing. The various ways resources are used provide a foundation for measurement for the different ways in the edge distributed devices in the various ways of empirical, monitoring, managing, and optimizing each of the

resource types. Given that edges operate under different types of variable load, the edges operate under telemetry, which is processed at different dead networks and at different heterogeneous embedded processing hardware constraints. This provides different ways to measure and evaluate the balance of computation, communication, resource utilization, and various processes.

3.2.1 Latency and Timing Metrics

The various types of performance metrics cover a core category of telemetry, which includes processing delays, task completion times, throughput, CPU, and even metrics of utilization, memory, and resource metrics. The metrics help measure and evaluate the ability of the edge node to execute and perform various workloads. The metrics measure and evaluate the distribution of the resources in the system, and the metrics of resources. The utilization of the resources in the system, and the distributed resources among the systems, is effective and is in a limited state of resources.

3.2.2 Resource Utilization Metrics

The network and communication parameters detail telemetry involving delays in transmission, packet loss, jitter, stability of the connection, and quality of the link. These parameters are critical in edge systems, as data must move through temporary and highly variable wireless channels. Tracking communication in a system makes it possible to select a new route, reconfigure the distribution of workload, or shift to a different edge node in order to maintain the continuity of the operation. Because network conditions are the primary factor that affects the latency, energy usage, and dependability of the distributed computation, communication-related telemetry is very important for maintaining a particular level of quality of service, as it is directly correlated to the level of service offered.

3.2.3 QoS/Anomaly Indicators

For telemetry related to security and anomaly, the primary focus is on curious system behavior, including the atypical invocation of system calls, unanticipated behavior or activity, or any of the patterns that are detached from normal operation. These symptoms may signal potential interruptions, resource underutilization, data manipulation, or other undesirable conditions that can compromise the reliability and security of the system. By continuously surveilling and evaluating these parameters, edge systems can issue warning messages, take remedial action, or contain affected components when the situation demands it. This type of telemetry is critical within distributed edge computing environments, where nodes are subjected to more risk.

3.3 Observability Scope

As shown in Fig. 2, the degree to which a monitoring system can be considered effective is heavily influenced by its Observability Scope, which is its boundaries of strategic focus that dictate what monitoring is done and what is filtered out as insignificant behavior. The heterogeneity of edge systems naturally makes a monitoring plan inadequate, as these networks comprise a great number of nodes and dynamic network conditions. By defining a scope on which a monitoring system focuses, it becomes possible to highlight data points and filter out noise that could overcome the edge system's limited hardware capabilities in a monitoring system.

3.3.1 Device-Level & Sensor-Level Monitoring

On the most comprehensive level, observability begins to concentrate on each individual node. This includes monitoring CPU cycles, memory pressure, and system calls to understand the translation of hardware events into system-level latency. In the context of edge nodes that need to run independently in various distant sites, observability is a lifesaver, and it helps to specifically isolate a problem of resource consumption or hardware decay that needs to be fixed prior to a local failure that could bring down a system.

3.3.2 Edge Node & Network-Level Monitoring

Network-level monitoring at the system as a whole, instead of focusing on just individual devices. It shows how different devices communicate with each other over the network. This helps in checking connection quality, delays in communication, and changes in performance caused by movement or unstable connections. Because edge devices work together and share data, a stable and reliable network is important. If network connections are weak or unpredictable, it can affect the entire system, causing service interruptions or uneven workload distribution.

3.3.3 Application/Service-Level Monitoring

Application-level observability correlates raw system data to what users are actually experiencing. It facilitates demonstrating how an application will perform under real-time conditions in terms of response time, accuracy and quality of results. Particularly in the domain of medical applications or industrial robotics, even minor latencies can cause big problems. This is to monitor when apps behave in the edge environment and, based on a policy, automatically allocate resources to meet performance requirements.

3.4 Architectural Layers

Fig. 2 categorizes architectural layers; the edge contains several layers, which range from small sensors at the bottom to fog nodes in the middle and a central cloud layer at the top. Each layer provides another perspective as to whether the system is functioning or not, and how trustworthy that layer might be. It is the combination of these layers that assists in determining where monitoring should take place, since not all monitoring can or should be executed from a battery-powered sensor, as it may not support the same degree of checking as a cloud server. So in this case, that monitoring becomes more flexible. It is able to easily detect problems at different layers and, at the same time, provide a high-level view that can reduce system circulation time.

3.4.1 Edge Layer

At the device level, this is the most basic layer of computing infrastructure. It is made up of IoT sensors, embedded devices, mobile clients, and lightweight processors that run close to the physical environment. At this layer, monitoring is done through hardware behavior, sensor readings, real-time actions, and system activity. This layer is critical for applications that demand immediate response, such as health monitoring, autonomous movement, and industrial sensing. Because of limited resources and changing operating environments, monitoring device level is critical for avoiding localized failures, increasing data integrity, and detecting abnormal patterns at this level without sending the problems to the upper level.

3.4.2 Fog Layer

The edge layer contains fog nodes, gateways, micro-data centers, and servers that actually process data at the point of origin. Instead of merely relaying information, these nodes filter, aggregate, and process data

as it flows through. To ensure this layer runs smoothly, we have oversight regarding resource usage, real-time behavior, and how well these nodes work with the cloud or other devices. At the end of the day, it is the edge that makes low-latency and mobile apps feasible. And by keeping monitoring, we can immediately spot bottlenecks, balancing work across the cluster to keep performance even when data flow gets unpredictable.

3.4.3 Cloud Layer

The powerful back end of the system is out in the cloud, tackling tasks that require a lot of storage and heavy processing. While edge devices address snappy, real-time operations, the cloud works on more holistic concerns, including retaining long-term data storage, working with complex models, and managing general security. At the cloud level, you're not making small instantaneous changes to adjust something immediately but rather tracking long-term patterns and ensuring data is kept synchronized as the system scales—and that the whole thing remains stable along the way.

3.5 Deployment Environment

A monitoring system does not exist in a vacuum but is affected by the world around it. This could be a paradigm for the development of the system itself because, irrespective of whether the system falls into the health sector, industry, or transportation network, the environment defines the conditions that must be met, as mentioned in Fig. 2. This could be the sensitivity of the information being dealt with or the allowable system downtime, because the moment these physical world conditions are factored in, the entire scenario of the system changes because of the monitoring challenges presented by the environment.

3.5.1 Kubernetes-Based Edge & Cloud Clusters

The Kubernetes edge system is useful in health monitoring platforms, biomedical sensing systems, and smart homes, in which reliable data collection is necessary. Most of the data in the above fields is sensitive, such as health data, critical alerts, and interactions between sensors, edge systems, and cloud systems. The key objective of health monitoring in these areas is accuracy of data, ensuring low latencies in critical alerts, as well as minimizing reliance on the central systems. The observability requirements suit various sensors, network connectivity, and changing workload requirements.

3.5.2 Containerized Edge Nodes

Industrial setups and smart grids are not merely about data acquisition and monitoring, but also ensuring that safety-critical systems are working perfectly without a hitch. In these setups, a network of edge nodes is embedded within factory floors and power lines, where any form of delay, no matter how minute, can result in a serious malfunction. In these setups that are already dealing with very intensive chores, such as real-time image analysis and malfunction detection, there is a high requirement on the monitoring tools to be extremely light and precise regarding their usage tracking.

3.5.3 Lightweight Edge Devices

Edge computing for transportation and smart cities is a domain that interacts with mobile clients, has fluctuating channels, and mixed workloads. It is used for traffic surveillance, vehicular analytics, and public safety surveillance, where low latency and location awareness pose a prominent challenge. System monitoring, in these domains, has to detect issues that may arise from mobility, a fluctuating Internet, coupled with a steady flow of data from all distributed nodes. Observability in these systems is, hence, focused

on monitoring latency, throughput, and mobility. The deployment context requires monitoring systems that are adaptive, lightweight, and capable of coordinating across wide geographic regions.

3.6 Observability Toolchain

Fig. 2 identifies observability toolchains as a key dimension, encompassing technologies such as eBPF, AI-driven analytics, distributed tracing frameworks, and telemetry pipelines. It is the combination of tools and techniques that help us make sense of the things that are happening across the edge. For edge computing, which is the combination of many different devices and unpredictable workloads, you cannot simply apply a single tool. It requires a blend of fine-grained understanding and the ability to scale. A reliable toolchain permits kernel-level tracing and real-time analytics, which endows the system with the situational awareness to recognize and mitigate issues early on, without introducing too much overhead to any of the processes.

3.6.1 eBPF-Based Tracing & Kernel Instrumentation

eBPF is the foremost technology for unprecedented granularity. It permits the observation of kernel activity, system calls, data packets, and OS interactions without the need to modify the source code of the targeted application. This is particularly advantageous to containerized platforms. Observability at the source allows us to identify and remedy bottlenecks and timing issues within a negligible duration. In a setting where every millisecond counts, eBPF is the indispensable technology to achieve optimal performance.

3.6.2 Artificial Intelligence and Machine Learning (AI/ML) Driven Analysis & Detection

In the systems that change rapidly, fixed rules and limits are not effective. This is where Artificial Intelligence and Machine Learning help. These technologies learn from system data to spot unusual behavior, predict possible failures, and improve how tasks are scheduled. Instead of waiting for a system to fail, machine learning allows problems to be detected early. This is especially important in areas like healthcare, where identifying a problem before it happens can prevent serious consequences.

3.6.3 Distributed Telemetry & Edge Analytics

It is apparent that the AI or ML techniques being utilized focus on the numerous ways that data patterns are inspected and examined within the context of tools of prediction and decision-making. It learns from data that is collected by telemetry with the purpose of examining the various ways that irregularity may occur. Another aspect of this technology is that it supports the various ways that adaptation can occur with tools of monitoring in an environment that lacks sufficient static threshold levels.

3.7 Architecture Paradigm

As presented in Fig. 2, the architectural paradigm focuses on the multiple potentialities found in the design of edge monitoring systems for the framework or structure, which considers how different responsibilities in systems for computing, communication, and observability are distributed across different devices, nodes, and cloud systems. Architectural paradigms found in edge computing provide the building blocks necessary for different means by which latency and fault tolerance can be traded off for monitoring depth. A taxonomic framework is further described with regard to different architectural choices in these paradigms because it explains different means by which different designs meet monitoring requirements and different means by which resource processes can be organized. These concepts are critical for determining different means by which monitoring frameworks meet the different gaps found in operations.

3.7.1 Edge–Cloud Hybrid

The edge-cloud hybrid model describes the different ways monitoring and computation are allocated among dispersed, localized nodes and centralized systems. This model describes the trade-off between rapid responsiveness and extensive computing power. It uses the edge for instantaneous filtering and local decisions and the cloud for prolonged storage and distributed optimization. These hybrid architectures offer different ways to advance flexibility and enhance data transmission. This division of responsibility supports the various ways frameworks adapt to the different types of variable workloads and network conditions.

3.7.2 Edge AI

Distributed Edge AI is the combination of different methods of machine learning and analytics at the edge. This paradigm provides different ways for systems to process and respond to data at the edge without the time delays of the global network. Monitoring within this paradigm focuses on and analyzes the different ways telemetry and application outputs interface to keep each of the AI processes within reliable bounds. Providing this type of intelligence creates different ways to augment autonomy and decrease the multiple types of communicative friction. This approach is especially valuable in the diverse needs of context-aware monitoring in the industrial and healthcare domains.

3.7.3 Micro Services

The containerized micro services paradigm focuses on how functionality is modularized into different isolated components. This approach aligns with the ways edge and cloud-native frameworks operate in concert for resource management and inter-service coordination. Monitoring in this paradigm is centered around the diverse ways container performance is monitored and the patterns of communication, or lack thereof, in each of the distributed services. The modular approach creates an environment for the different ways fault isolation and dynamic scaling, or the lack thereof, are achieved. This makes it appropriate for the many different ways complex edge deployments need to scale.

4 Review and Comparative Analysis of State-of-the-Art Edge Monitoring Solutions

This section comprises two subsections. In Section 4.1, we discuss the state of the art monitoring solutions, and in Section 4.2, we discuss about the comparative analysis of the current state-of-the-art monitoring solutions using the thematic taxonomy we have developed in Section 3.

4.1 State-of-the-Art Edge Monitoring Solutions

Edge computing is a paradigm that has emerged to address the challenges posed by traditional cloud computing architectures. This taxonomy synthesizes insights from research papers to provide a broad overview of the challenges, solutions, applications, evaluation methodologies, integration with emerging technologies, and future directions in the field of edge computing.

4.1.1 Foundational and Architectural Edge Frameworks

This category includes solutions that define the architectural foundations of edge computing, focusing on workload distribution, latency reduction, and integration with cloud environments. These works provide the underlying infrastructure and system design principles upon which monitoring and observability mechanisms are built, but they do not explicitly implement advanced monitoring techniques.

a. EC's Rise

In [31], Satyanarayanan discusses the different ways in which edge computing has evolved as a new technology to solve the issues of outdated cloud systems. The attention is drawn to the measurements of different manners in which massive amounts of data are governed in the Internet of Things (IoT). This evolution of placing compute and storage nodes, also known as "cloudlets," closer to the data source offers a basis for different ways in which latency is overcome and privacy is ensured. The different manners in which real-time tracking can be supported by cloud computing systems are offered by different methods involving bandwidth and latency. This approach offers different manners in which sensitive data can be preprocessed by sensor data which is an important solution in addition to cloud systems to ensure different manners in which agility and efficient data management.

b. EC Emergence

In [32], the need for edge computing has been discussed in terms of it becoming a revolutionary technology that brings resources closer to different types of mobile devices as well as sensors. This technological change provides different means of efficiently achieving scalable cloud services for IoT. The different means by which the application workflow of cloudlets relies on edge analytics to improve bandwidth as well as the efficiency of the system have been described. Because of research work done by different research initiatives, edge computing stands ready to provide the means by which processing power gets closer to the end-user.

c. Secure IoT EC

In [12], Wang et al. introduce a secure IoT service architecture, which targets the different ways the computation workload is distributed across cloud and edge nodes. The proposed architecture targets the measurement of the different ways the network workload and sensitive tasks are distributed, with a focus on balancing tasks with a high latency rate at the edge nodes. The proposed system also targets the integration of all the different ways the balancing algorithms and cooperation methods ensure the optimal system functionality related to efficiency and security. The system enables different ways of improving system responsiveness and overcoming different ways of communication delays. The system uses all the different ways of ensuring data privacy and reliability.

4.1.2 System-Level Observability and Lightweight Monitoring

This group focuses on efficient, low-overhead monitoring mechanisms designed for resource-constrained edge environments. It includes kernel-level observability (eBPF), real-time performance profiling, telemetry aggregation, and container-level monitoring. These solutions emphasize scalability, real-time data collection, and minimal system overhead, making them suitable for large-scale and dynamic edge deployments.

a. eBPF Cloud Apps

In [38], Fournier et al. review the different ways in which eBPF innovation is utilized inside cloud administrations, and underline its role in the numerous ways by which assets are arranged and secured. This study has sorted use cases of eBPF under various zones: networking, storage, and security, among others, to feature the different ways it has been embraced to accomplish execution changes. It gives a premise for the different ways by which analysts look at arrangement methods and execution increases. Various ways to seek machine learning and programmable storage have likewise been recognized through eBPF. On-path vs. off-path monitoring for different structure plans gave insight into the various ways in which complexity is estimated and tried in an execution.

b. MessTool Real Time (RT)

In [33], Gowtham et al. discuss the various ways real-time capabilities are achieved on edge servers running on Linux through the MessTool framework. This tool focuses on the measurement of the various ways industrial applications perform on edge nodes, providing a foundation for detailed insights into timing performance. The framework utilizes the various ways of eBPF probing for precise measurement and focuses on the different ways time-critical applications are profiled. It addresses the various challenges in monitoring computing nodes and the different types of remote I/O controllers. The architecture includes various components like the Monitoring Manager and Aggregator to handle the different ways distributed measurements are processed. This provides various ways to evaluate real-time monitoring within the hardware constraints of edge environments.

c. EC eBPF Observability

In [34], a novel eBPF-based monitoring architecture is presented. Such a system affords edge observability at a fine grain. This approach leverages eBPF to observe kernel-side events and resource utilization without added overhead on the target system. This enables the system to gather data with minimal latency and performance overhead, making it suitable for a low-power edge network. The architecture also enables adaptive tracing and continuous monitoring of system health, aiding in increasing reliability. Our test demonstrates that the system incurs a minimal amount of CPU overhead and maintains an event capturing precision at the microsecond scale.

d. EC Lightweight Telemetry

In [39], a distributed lightweight telemetry system is proposed to improve the various ways real-time observability is achieved in large-scale edge environments. The framework focuses on the measurement of the various ways local monitoring agents adaptively collect and compress system performance data. This provides a foundation for the various ways summarized metrics are forwarded to edge collectors while maintaining a limited state of resource utilization. In contrast with typical cloud-based telemetry, this architecture enables various kinds of hierarchical aggregation, such that the variety of real-time insights is kept even if constrained by hardware. Experimental results show how it is possible to lower the transmission latency and computational overhead, confirming that the system ensures accurate monitoring between edge networks in various ways.

e. EC Orchestration Monitoring

In [40], a microservice-oriented orchestration monitoring system is developed to integrate the various ways monitoring capabilities work within Kubernetes environments. The framework tracks the various ways container lifecycles and inter-service communications are measured, providing a foundation for the different ways resources are managed. Through the various ways of continuous observability, the system provides different ways to predict anomalies and enable proactive fault mitigation. This highlights the various ways orchestration-aware monitoring is essential for supporting the different types of scalable and autonomous edge infrastructures.

4.1.3 Application-Specific and Domain-Oriented Monitoring

These solutions are designed for specific application domains such as healthcare, smart infrastructure, and intelligent transportation systems. They leverage edge computing to enable real-time data processing, reduce latency, and optimize bandwidth usage. While highly effective within their domains, these approaches are often not generalized for broader edge environments.

a. EC Smart Grids

In [41], the implementation of an edge computing framework is discussed in relation to different aspects of real-time monitoring in smart grid systems. It points out that real-time monitoring in smart grid systems takes place in different forms that are measured by the different aspects in which traditional real-time monitoring systems have shortcomings in power grid systems. The significance of different aspects in which edge computing enhances frame rate and lowers detection delay compared to cloud computing systems has been mentioned. Various aspects of solving the problem related to scheduling of monitoring systems and edge server systems have been explained in relation to designing an algorithm for an efficient solution. The different aspects in which the framework operates in smart grid systems have been explained by different experiments.

b. Insulator Detection

To improve the accuracy and responsiveness of this detection system of power grid inspection, an intelligent online monitoring system for insulator self-explosion is proposed in [37]. This system integrates deep learning and edge computing. The framework deploys edge nodes to process captured Unmanned Aerial Vehicle (UAV) image data locally before sending summarized results to the cloud. It uses lightweight Single Shot Detector (SSD) neural networks for feature extraction and classification of insulator defects. When compared to conventional cloud-only solutions, this architecture dramatically lowers computational costs and transmission latency. Additionally, the system uses distributed edge servers to manage parallel image inference tasks, increasing network reliability and fault detection effectiveness. The proposed edge-deep learning hybrid system thus represents a scalable and efficient monitoring framework for power grid asset management, supporting autonomous maintenance and fault prediction.

c. Traffic Monitoring

Tang et al. [35] focuses on improving performance and responsiveness in such distributed systems using edge computing, in a hybrid edge-cloud setup with tasks running on both edge nodes and the cloud. Local processing, for tasks like motion detection and object tracking, takes place on an edge node. Longer-term storage and advanced processing tasks are carried out in the cloud. Edge processing serves to save bandwidth and reduce latency compared to cloud-only systems by eliminating the step of uploading video to the cloud for processing. This leads to a more uniform load distribution across the network, allows for more rapid data acquisition, and permits real-time feedback. These properties are corroborated by empirical studies showing that the framework is scalable and responsive to changing conditions, e.g., the time to download the data falls considerably when varying the sampling precision.

4.1.4 QoS-Aware, Mobility-Aware, and Performance Monitoring

This category focuses on maintaining system performance and service reliability under dynamic conditions. It includes QoS monitoring, mobility-aware frameworks, and fault analysis mechanisms. These solutions aim to ensure stable service delivery by monitoring latency, throughput, and system resilience, particularly in environments with fluctuating workloads and user mobility.

a. QoS Monitoring

Zhang et al. [42] proposed a novel approach called ghBSRM-MEC (graph-based Bayesian Service Relationship Model for Mobile Edge Computing) that focuses on monitoring Quality of Service (QoS) in the context of mobile edge computing. This approach tackles challenges related to user mobility and interdependencies among QoS parameters with the goal of minimizing monitoring inaccuracies. It proposes the idea of forming parent attributes to mitigate dependence between QoS attributes and switching edges in a dynamic way for monitoring purposes. The architecture of ghBSRM-MEC, the algorithmic formulation

of the methodology, and the empirical assessment of the methodology include difficulties within traditional QoS monitoring, the importance of probabilistic monitoring approaches, and the need for effective QoS monitoring in mobile edge computing.

b. Fault Impact

In [43], they examine the effect that various resource-contending faults have on the latency of edge-computing processes. And also, it looks into multiple overloads inflicted, for instance, on object recognition and speech recognition applications. The goal is to find out which faults are the worst for meeting the requirements to improve fault tolerance on edge computing systems. The research method involves benchmarking applications on a Raspberry Pi computing device and then performing a fault injection experiment to measure the disruption's impact on the performance metrics, which is latency. The results of the study reveal resource-stressing edge faults that can be addressed to increase the trustworthiness of edge systems and systems for resourcedeficient area.

c. EC Fault Resilience

In [44], a fault-resilient edge framework is proposed to analyze the various ways system overloads and hardware faults impact application latency and availability. The study introduces a foundation for measurement through fault injection mechanisms that emulate the various ways CPU and memory stress occur at runtime. By profiling the various ways different fault types affect the system, the research identifies different ways to evaluate critical vulnerabilities in time-sensitive applications. Experimental results show how the various ways of dynamic load redistribution can recover lost performance after transient failures, providing various ways to enhance the reliability of self-healing edge infrastructures.

d. EC QoS Mobility

In [36], a mobility-aware QoS monitoring framework is proposed to ensure the various ways consistent service delivery is managed in mobile edge scenarios. The system continuously monitors the various ways latency, jitter, and throughput interact as users move between network cells. By using probabilistic modeling, the framework reduces errors in monitoring when users or devices are moving. This helps limit service disruptions and keeps the quality of service more stable. It results in smoother and more reliable connectivity, which is especially important in smart transportation systems.

4.1.5 AI-Driven and Intelligent Monitoring Systems

This group represents the integration of artificial intelligence into edge monitoring. These solutions enable intelligent anomaly detection, model performance tracking, and adaptive system behavior. They mark a transition toward autonomous and predictive observability, where monitoring systems can dynamically respond to changes in workload and data patterns.

a. EC ML Drift Detection

In [45], a monitoring solution for machine learning drift detection is implemented to focus on the various ways inference accuracy is sustained at the edge. The system tracks the various ways data distribution shifts and prediction errors are measured at edge nodes to detect early signs of model degradation. Upon detection, the framework provides various ways to trigger retraining workflows and synchronize models with the cloud. This approach ensures the various ways distributed edge ML models remain adaptive, improving the different ways decision reliability is maintained in dynamic and data-intensive scenarios.

b. Federated EC Privacy

In [46], a federated edge monitoring system is proposed that enables secure data analysis in distributed IoT settings, preserving the user's privacy. Sensors spot trouble locally and send only encrypted data to

a centralized system. This method is used to protect sensitive information. By integrating encryption and secure aggregation, the system provides strong data confidentiality. The series of experiments performed to confirm that the proposed secure monitoring technique works successfully with a marginal accuracy drop and lower communication.

c. EC Dynamic eBPF-driven Syscall Filtering and Anomaly Mitigation (DeSFAM)

Zehra et al. [28] proposed DeSFAM, an adaptive monitoring framework designed to improve the various ways real-time observability and threat mitigation occur in containerized environments. By combining the various ways of AI-driven anomaly detection with eBPF-based kernel tracing, the system allows the monitoring layer to adapt to the various ways workloads shift. Experimental evaluations show the various ways DeSFAM achieves high detection accuracy in identifying malicious activities while reducing the various types of monitoring overhead. The framework provides various ways for edge-hybrid environments to outperform static systems in terms of the measurement of latency and adaptability.

4.1.6 Other Monitoring Solutions

Several edge monitoring solutions have been extensively discussed in various research papers, each addressing different aspects of edge computing environments. Astrolabe, for example, is a hierarchical monitoring system that organizes nodes into domains (zones) and uses on-the-fly aggregation to compute summaries of system data, enhancing scalability and efficiency [47,48]. There are a number of systems available to manage a large, complex environment. SDIMS [49] (Scalable Distributed Monitoring System) provides a robust infrastructure for large network systems and distributed resources, ensuring effective monitoring across the networks. Fog/Edge Monitoring emphasizes scalability, non-intrusiveness, and locality. They often feature data aggregation, event-driven monitoring, and long-term storage capabilities that support fog and edge infrastructures [50–52]. Cloud monitoring solutions specially designed for cloud environments that mainly focus on scalability, robustness, non-intrusive, and real-time monitoring to maintain smooth cloud operations [53,54]. IoT monitoring solutions are designed for Internet of Things (IoT) devices, which are sensors and smart devices [55–58], prioritizing real-time monitoring, data processing, and context awareness to handle the unique challenges of IoT ecosystems [59]. Some monitoring solutions are designed for applications that can adapt automatically. These systems track performance and resource usage to ensure applications run efficiently. Additionally, in [60–63], these solutions focus on cloudlets, which are small servers placed close to users to reduce delays. Monitoring in these environments emphasizes fast response times and scalability. The influence of monitoring in fog and edge computing [6,64,65] underscores the necessity for scalable, non-intrusive, and context-aware monitoring solutions. Moreover, holistic monitoring services for fog/edge infrastructures propose comprehensive solutions that integrate these critical features [32,66]. Recent advancements highlight the integration of AI-driven techniques into monitoring systems for complex distributed environments. Machine learning enhances anomaly detection and log analysis and reduces alert noise in Kubernetes-based systems. Additionally, AI-driven microservices monitoring frameworks leverage fine-grained telemetry and model-centric metrics to improve system visibility, root cause analysis, and overall observability in dynamic cloud-native environments [67,68]. Lastly, discussions on push vs. pull approaches in web-based network management highlight the importance of monitoring and control in optimizing edge computing environments [29,41]. Collectively, these solutions illustrate the diversity and complexity of edge monitoring, emphasizing the crucial need for adaptable and efficient monitoring frameworks to support the dynamic nature of edge computing.

The discussion of state-of-the-art edge monitoring solutions highlights a wide range of architectures, tools, and application-specific implementations, each addressing different challenges in edge environments. Due to the observation that reported numbers from various works are obtained from different platforms with

different datasets, workloads, and edge/fog/cloud deployment scenarios, direct comparison of these numbers may not be completely apples-to-apples. Thus, comparisons in this paper are conducted based on a common set of axes (for example, latency, overhead, network bandwidth consumption, detection accuracy) when available. This comparison will serve as a guideline to provide relative and trend analysis among existing approaches, based on both quantitative numbers and architectural qualitative analysis. While these studies provide valuable insights into design choices and practical deployments, a deeper understanding requires an analytical comparison across common dimensions. To achieve this, the following analysis evaluates these solutions using a unified framework, enabling a clearer identification of their strengths, limitations, and overall effectiveness in supporting scalable and reliable edge observability.

4.2 Comparative Analysis of State-of-the-Art Edge Monitoring Solutions

This sub-section provides a critical analysis of edge monitoring solutions based on the parameters set in the taxonomy in Section 3. This comparison is categorized into qualitative and quantitative aspects of existing edge monitoring solutions. The monitoring solutions presented in the table were analyzed, and their distinguishing attributes and features are illustrated in Table 2.

Edge monitoring is taking charge of modern distributed systems in terms of reliability, responsiveness, and security. As edge computing spreads into areas like healthcare and industrial automation, monitoring systems need to handle a wide range of devices while also working within limited resources. To provide a unified understanding of how research approaches this, this section presents a detailed analysis of the various ways state-of-the-art frameworks are built. The analysis synthesizes how each solution uses telemetry, architectural patterns, and monitoring goals to achieve system visibility.

Across the surveyed literature, the various ways monitoring solutions demonstrate their primary intent are quite clear. Several frameworks focus on the measurement of the various ways performance is optimized by reducing response latency and balancing computation. General studies emphasize the different ways proximity-based computation lowers delays in various latency-sensitive domains like vehicular networks [31,32]. More targeted systems leverage the various ways edge-side deep learning minimizes transmission delays [37]. While other tools, such as MessTool, focus on the various ways real-time responsiveness is measured through nanosecond-level timing [33,35]. Security-centric frameworks expand this intent by focusing on the various ways reliability and threat mitigation are handled through risk assessment and dynamic balancing [12] distributed IoT deployments, while eBPF-based adaptive systems strengthen detection accuracy and minimize attack surfaces through kernel-level observability and anomaly identification [28,34].

The variability in how telemetry is acquired illustrates the various ways methodological diversity exists. eBPF-based systems prioritize the various ways kernel-level telemetry captures system calls, and packet flows to provide a foundation for measurement with minimal overhead [33], distributed observability [34], and adaptive threat mitigation [28]. Conversely, deep learning frameworks in smart grids or manufacturing rely on the various ways sensor-level signals, such as image streams or voltage patterns, are processed locally [35,37]. Health-oriented systems monitor the various ways biomedical signals support real-time diagnosis [32]. Different monitoring approaches focus on different goals. Some QoS-driven systems mainly track network metrics such as delay and jitter to check whether the network is stable [36]. This shows that data collection is usually designed to match the specific needs of each application domain.

Table 2: Comparison of state-of-the-art monitoring solutions.

Paper Reference	Monitoring Intent			Telemetry Indicators			Observability Scope			Architectural Layers			Deployment Environment			Observability Toolchain			Architecture Paradigms		
	PO	RTR	SRA	LTM	RUM	QAI	DLSM	ENNM	ASLM	EL	FL	CL	KECC	DCEN	ELED	EBT	AIMD	DTEA	HEC	EAI	MS
EC's Rise	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
EC Emergence	✓	✓	✗	✓	✓	✗	✓	✓	✓	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✗	✗
Secure IoTEC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗
eBPF Cloud Apps	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗
MEC Smart Grids	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗
MestTool IRT	✓	✓	✗	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✓	✗	✗	✓	✗	✗	✗
QoS Monitoring	✓	✓	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓
Insulator Detection	✓	✓	✗	✓	✗	✓	✓	✓	✗	✓	✗	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
Traffic Monitoring	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✓	✗	✗
Fault Impact	✓	✗	✓	✓	✓	✗	✓	✓	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗	✗	✗
Lightweight Telemetry	✓	✓	✗	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	✗
EC eBPF Observability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗
EC Fault Resilience	✓	✗	✓	✓	✓	✗	✓	✓	✗	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✓
EC ML Drift Detection	✓	✗	✗	✓	✗	✓	✗	✗	✓	✓	✓	✗	✗	✓	✗	✓	✗	✗	✓	✗	✗
EC QoS Mobility	✓	✓	✗	✓	✗	✓	✗	✓	✓	✓	✓	✓	✗	✗	✗	✓	✗	✓	✗	✗	✗
Federated EC Privacy	✗	✗	✓	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗	✗	✓
Orchestration Monitoring	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✗
EC DeSFAM	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓

Note: **PO**: Performance Optimization, **RTR**: Real-Time Response, **SRA**: Security Risk Assessment, **LTM**: Low-Level Telemetry Metrics, **RUM**: Real User Monitoring, **QAI**: Quality of Analytics Insights, **DLSM**: Device-Level System Monitoring, **ENNM**: End-to-End Network Monitoring, **ASLM**: Application & Service-Level Monitoring, **EL**: Edge Layer, **FL**: Fog Layer, **CL**: Cloud Layer, **KECC**: Kubernetes Edge Cloud Computing, **DCEN**: Data Center Environment, **ELED**: Edge-Level Embedded Devices, **EBT**: Edge-Based Telemetry, **AIMD**: AI-Driven Monitoring & Diagnostics, **DTEA**: Distributed Trace & Event Analysis, **HEC**: Hybrid Edge Computing, **EAI**: Edge AI, **MS**: Microservices.

A comparison of monitoring is done to cover the various ways of observability that are achieved across different layers. Some solutions focus on device-level monitoring, where raw data from sensors is processed; this would be placed as close to the device as possible [13,37]. In contrast, other frameworks focus on the different ways of edge-node observability that capture metrics from the operating system and kernel [31,33]. There are also hybrid systems that monitor application behavior and service interactions across multiple layers [35]. Together, the approach in [36] highlights the need for monitoring all layers at the same time to stay accurate in changing environments.

Many systems also use layered architectures to spread computation efficiently. For example, smart-grid systems often use a hierarchy where edge devices handle quick analysis, and the cloud requires heavier processing [35]. Healthcare monitoring systems for elderly users use multistage pipelines that send data only when an unusual change is detected. eBPF-based systems focus on monitoring directly at the edge to gain detailed system insights without relying on other nodes [28,34].

Mobility-aware systems mainly monitor the network layer to keep services stable as users move [36]. The differences among these approaches demonstrate how architectural choices are shaped by latency demands, resource constraints, and domain-level requirements. Deployment environments also vary widely. Healthcare systems often rely on embedded devices and smart-home nodes to reduce dependence on the cloud [12]. Vision-based uses powerful GPU-enabled devices to accelerate real-time detection [35,37]. eBPF-based monitoring systems typically run on Linux systems that support high-performance data collection and observability [28,31,32,34]. These factors illustrate the different ways edge architectures must adapt to the various types of hardware constraints and domain-specific requirements.

Finally, monitoring tools and architectural designs differ across systems. eBPF-based tools focus on low-overhead monitoring by working at the kernel level [28,33,34]. AI-based approaches use compressed models so they can run on limited hardware. Despite these strengths, there are various ways limitations remain, such as the lack of end-to-end observability across all layers. These gaps reveal the various ways unified and security-aware frameworks are still needed to provide multi-layer visibility in the different types of resource-constrained edge environments.

Evaluation of Observability Solutions: Tools, Features, and Performance Metrics

Evaluating existing monitoring and observability solutions is important for understanding how well they work in real systems. How they performed in the real world and how suitable they are for modern edge-based microservices. Previous research has introduced many solutions that focus on areas such as performance monitoring, detecting anomalies, identifying faults, and adapting to changing conditions across edge and cloud environments. However, these solutions differ widely in terms of goals, the type of data they collect, the technologies they use, and how thoroughly they are evaluated. To clearly highlight these differences, a qualitative comparison of key edge monitoring and observability solutions is shown in Table 3.

Table 3: Qualitative comparison of edge monitoring solutions.

Paper Reference	Observability Purpose	Data Evaluated	Technology Used
EC's Rise	PA, AH	Metrics, latency trends	Edge-cloud architecture, virtualization
EC Emergence	PA, AH	Metrics, system behavior	MEC architecture, distributed systems

(Continued)

Table 3 (continued)

Paper Reference	Observability Purpose	Data Evaluated	Technology Used
Secure IoTEC	AD, RCA	Logs, security events	IoT security frameworks, encryption
eBPF Cloud Apps	PA, AD	Traces, syscall data	eBPF, Linux kernel
MEC Smart Grids	PA	Sensor data, grid metrics	Smart grid IoT, edge analytics
MestTool IRT	PA, AD	Fault traces, metrics	Real-time fault detection algorithms
QoS Monitoring	PA, RCA	QoS metrics	QoS models, MEC
Insulator Detection	AD	Image data	Computer vision, edge AI
Traffic Monitoring	PA, AD	CV event data	Computer vision, edge processing
Fault Impact	PA, AD	Impact metrics	Edge analytics
Lightweight Telemetry	PA	Telemetry metrics	Lightweight probes
EC eBPF Observability	PA, AD	Syscalls, traces	eBPF, kernel tracing
EC Fault Resilience	PA, RCA	Fault logs, impact metrics	Resilience modeling frameworks
EC ML Drift Detection	AD	Model drift metrics	ML models, drift analysis
EC QoS Mobility	PA	Mobility metrics	MEC mobility models
Federated EC Privacy	RCA, AD	Federated data	Federated learning, privacy models
Orchestration Monitoring	PA, RCA	Orchestration logs/metrics	Docker/K8s orchestration
EC DeSFAM	PA, AD, RCA	Syscall traces	eBPF, VAE + Isolation Forest

Note: PA = Performance Analysis, AH = Anomaly Handling, AD = Anomaly Detection, RCA = Root Cause Analysis.

This comparison focuses on the various ways the primary observability purpose, the types of data evaluated, and the technologies employed interact within each approach. As shown in the synthesized data, early edge-centric studies primarily emphasize the various ways performance analysis and architectural understanding are achieved through the measurement of latency trends, system behavior, and resource utilization. These works largely rely on the various ways edge–cloud architectures and distributed systems models address scalability and the different challenges of the various ways observability requirements change depending on the different application contexts.

Advanced approaches further integrate the various ways machine learning models and privacy-preserving mechanisms support the different processes of adaptive monitoring. Overall, the research highlights a clear progression from the various ways coarse-grained monitoring was once handled toward the different ways fine-grained, intelligent, and application-aware observability is achieved today. Following this qualitative assessment, a quantitative performance comparison of the various ways state-of-the-art solutions perform is provided in the subsequent analysis in Table 4.

This analysis examines the various ways key metrics such as latency, CPU and memory overhead, and bandwidth consumption are measured across different frameworks. As illustrated in the performance data, kernel-level instrumentation techniques provide a foundation for the various ways microsecond-level latency is achieved with minimal performance overhead. This makes them highly suitable for the various ways real-time observability is maintained in resource-constrained edge nodes. These monitoring frameworks demonstrate the various ways latency is significantly reduced compared to cloud-only pipelines by performing the various processes of local processing and inference directly at the edge, thereby lowering communication overhead.

Solutions that integrate the various ways of artificial intelligence exhibit high detection accuracy and provide different ways for effective anomaly identification while maintaining a limited state of resource utilization. In particular, the various ways lightweight inference models and adaptive security mechanisms are combined demonstrate a favorable balance between the measurement of detection performance and computational cost.

However, the quantitative results also reveal the various ways many conceptual and architectural studies lack the necessary empirical benchmarking, which limits the different ways they can be applied to performance-critical deployments. Furthermore, some solutions report the various ways strong accuracy metrics are achieved but require specialized hardware or kernel-level access, which can restrict the various ways portability and scalability.

Together, these findings present a more comprehensive assessment that integrates the many ways in which qualitative architectural insights and quantitative performance evidence complement each other. As a side effect of this double analysis, we demonstrate the nature of different kinds of limitations in state-of-the-art approaches, and also peel off how large-scale validation and multi-node evaluation are still not enough. These gaps provide a foundation for the need for an observability framework that delivers the various ways of fine-grained visibility and adaptability across the different types of edge microservice environments.

Additionally, domain-specific solutions—including smart grids, traffic monitoring, and healthcare that demonstrate.

Table 4: Quantitative comparison of edge monitoring and observability solutions.

Paper	Latency/Response Time	CPU/Memory Overhead	Bandwidth/ Transmission	Accuracy/Detection Metrics	Other Metrics/Notes
Demystifying eBPF Performance	Microsecond-level syscall tracing; sub-microsecond observation latency	CPU overhead typically < 5%	N/A	N/A	High-precision kernel instrumentation
Real-Time Monitoring & Analysis (MessTool IRT)	UDP delay: 18–44 µs; Modbus-TCP RTT < 2 ms; kernel array read 3.9 ns; syscall 485 ns	N/A	N/A	N/A	Provides nanosecond-level measurements
Mobility & QoS Monitoring (ghBSRM-MEC)	Classifier training time: ≈ 2.17 s for 2000 samples	N/A	N/A	Reduces QoS-related monitoring inaccuracy	Uses probabilistic ratio + KNN fallback
Edge Computing Vision & Challenges	N/A	N/A	N/A	N/A	Conceptual paper; no numerical benchmarks
Smart Traffic Monitoring System	Real-time responsiveness; edge inference reduces latency vs. cloud-only	N/A	Bandwidth greatly reduced (only processed features sent)	High detection precision reported	Edge–cloud hybrid architecture
Real-Time Edge Computing Framework	N/A	N/A	N/A	N/A	Provides architectural guidelines
DeSFAM (AI + eBPF Adaptive Security)	Enforcement latency: 0.540 µs; policy updates 65–82 µs	CPU + 1.74%; memory reduced ~6.1% (≈50–55 MB)	Monitoring cost reduced 32%	Accuracy 96%; F1 0.92; Precision 94%; Recall 90%; blocked 14/14 CVE attacks	Syscall reduction rate: 78.6%
Ultra-Lightweight Co-Inference Model	Low detection latency; training time few seconds per epoch	Compression ratio 0.341; quantized model runs at 6–8 bits	N/A	F1: 0.88–0.94 (dataset-dependent)	Works on extremely constrained devices
Insulator Self-Explosion Monitoring	Edge inference latency ≈ 30 ms (lightweight SSD model)	Edge device power: 25.6 W; standby 0.5 W	Large reduction due to local inference	Accuracy 95.75%; VAL 94.68%; AUC 0.9153	Uses NVIDIA Jetson TX2
eBPF Edge Node Real-Time Capability	Kernel tracing latency in microseconds	Almost zero overhead; minimal syscall perturbation	N/A	N/A	Shows eBPF suitability for real-time edge tasks
Emergence of Edge Computing	N/A	N/A	N/A	N/A	Conceptual; describes ecosystem trends

(Continued)

Table 4 (continued)

Paper	Latency/Response Time	CPU/Memory Overhead	Bandwidth/ Transmission	Accuracy/Detection Metrics	Other Metrics/Notes
Resource-Stressing Fault Analysis	Reports application latency variation under fault injection	N/A	N/A	N/A	Identifies worst fault types for workloads
Secure IoT Service Architecture (Cloud + Edge)	N/A	N/A	Communication reduced through hybrid routing	N/A	Provides secure multi-layer architecture
Extended BPF: Application Perspective	N/A	N/A	N/A	N/A	Conceptual paper on eBPF; no numerical evaluation

5 Open Research Issues and Challenges in Edge Computing

Edge computing is a speedily growing field with significant potential to transform the way data is handled and delivered. However, it also presents numerous research challenges and issues that must to be addressed for its effective execution and optimization. Here are some of the key open research issues and challenges in edge computing.

Fig. 3 presents a consolidated view of the major open research issues and challenges in edge computing identified through the literature review and analytical evaluation conducted in this study. The diagram highlights six interrelated challenge domains, including comprehensive observability across edge layers, unified telemetry collection and cross-layer data fusion, scalability and real-time processing constraints, adaptability to mobility and dynamic edge environments, privacy and security challenges across edge layers, and emerging research opportunities and future directions. These challenges reflect the complexity of edge computing systems, which are characterized by various challenges such as heterogeneity, decentralization, and strict latency requirements. The conceptual roadmap provides a foundation for the various ways each challenge is examined in depth, particularly regarding its impact on the different ways edge observability and performance monitoring are evaluated across smart scheduling mechanisms.

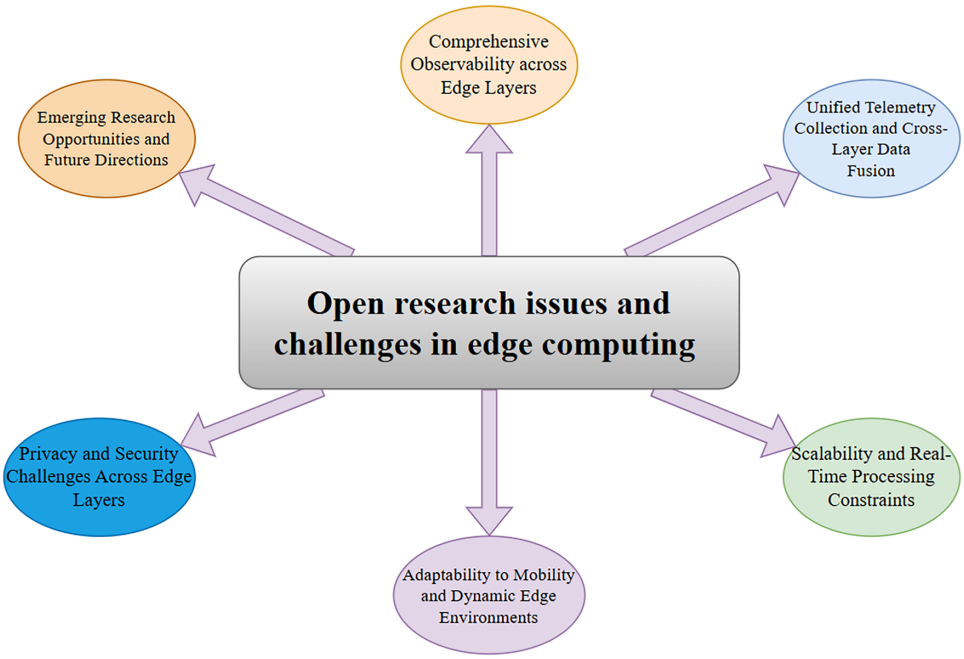

Figure 3: Key open research challenges in edge computing include observability, telemetry, scalability, mobility, and security.

5.1 Comprehensive Observability across Edge Layers

Even though edge computing approaches have emerged, the problem of achieving end-to-end visibility has proven to be difficult. To date, much of the research is siloed, looking at the health of the system at the infrastructure layer or monitoring the cloud, rather than providing a unified view over the entire data flow, as performance problems are rarely contained within a single component. These problems could be at the operating system, network, or application layer. To detect problems in real time, a monitoring framework should be multi-layered and should be able to analyze and reason on real-time information

from all layers [69]. At the same time, observability should be provided without overloading resource-constrained edge devices, which are typically battery-powered and computationally limited. Solving this problem would enable accurate observability for mobile edge applications without sacrificing performance and battery lifetime.

5.2 Unified Telemetry Collection and Cross-Layer Data Fusion

Edge data may include low-level telemetry streams such as kernel traces, but also high-level semantics such as mobility. Due to the wide variety in the format and granularity, we often see these monitoring pipelines become siloed, which limits their usefulness in analyzing the edges. Without a common way to collect and fuse this data in real time, accurate root cause analysis and diagnosis of small performance drifts remain beyond the capabilities of current tools. Future research needs integrated architectures that can fuse this multi-source data in real time to address the limitations of current solutions [54,69]. We need a platform that reliably collects data and has the ability to correlate data to see where jitters or delays in scheduling or processing are occurring.

5.3 Scalability and Real-Time Processing Constraints

As edge deployments grow in scale, observability systems face a tradeoff between the telemetry emitted by the edge nodes and resource consumption on each node, which is typically very low. Many applications that require real-time monitoring have strict latency requirements that are not compatible with customary monitoring approaches that are too "heavy". The problem is that the high frequency and accuracy of the trace, i.e., monitoring, can interfere with the task's performance itself [40]. To solve this problem, ultra-low-overhead observability with sufficient precision, without sacrificing responsiveness, is likely needed. Adaptive sampling and event-driven tracing appear to be the most promising approaches in this regard. Thus, frameworks like this could be used to provide high granularity at high load while imposing little or no overhead when the system is not stressed.

5.4 Adaptability to Mobility and Dynamic Edge Environments

Mobility may add a type of uncertainty that makes it difficult to measure network latency and link stability. Most of the existing solutions assume some other kind of stable condition, and therefore are not able to quickly adapt to mobility [70]. This entails considering diverse mobility-aware observability formulations that combine real-time and predictive perspectives, as well as considering various forms of proactive changes in the scheduling and tracing depth relative to the different forms of changes in the environment.

5.5 Privacy and Security Challenges across Edge Layers

The deeper the observability mechanisms are, the more types of interaction involving privacy and security issues have to be considered. The telemetries collected from the devices can reveal the different types of sensitive user activities and location patterns, which call for the various types of strong privacy-preserving mechanisms. Different types of instrumentation techniques that involve the kernel can shed more light on different types of attack surface, which can be addressed by developing different types of secure observability frameworks with encrypted transport, as well as by employing different types of anonymization to maintain the different types of analytical value [29,38].

5.6 Emerging Research Opportunities and Future Directions

Aside from the above-mentioned challenges, there are many intersections between observability and mobility-aware edge computing, such as self-adaptive observability systems and telemetry control systems

that are able to follow dynamic workloads. Interestingly, a future research direction can include investigating the type of causal relations that exist over an entire pipeline and the performance anomaly metrics these relations influence. Future work could additionally consider the core observability and type of schedulers used, adaptive task offloading, as well as resource allocation achievable.

6 Conclusion

Edge computing is increasingly becoming a core paradigm for latency-sensitive and data-intensive applications. Its proximity to the end users presents unique monitoring and observability challenges in guaranteeing system reliability over a highly distributed and heterogeneous environment. In contrast to homogeneous, elastic infrastructures underlying conventional cloud-centric approaches, edge environments are characterized by limited computational resources, limited energy resources, shared and constrained networking resources, and stringent real-time guarantees. In those regards, a conceptualization of monitoring and observability in the context of the edge is described.

This work provides a thorough taxonomy of the state-of-the-art edge monitoring and observability solutions based on their optimization objectives, observability scope, telemetry characteristics, and deployment architectures. The former is characterized by tracing the source code at the kernel level with relatively low overhead, and the latter is typified by learning-based observability that is semantically richer but computationally more expensive. The differences between existing solutions emphasize the need for dynamic observability frameworks that consider edge resource constraints.

From the comparative analysis have been identified in current research. The comparative analysis reveals a notable number of solutions and proposals that focus solely on one aspect, either the effectiveness of anomaly detection or performance monitoring, without cross-layer visibility or a holistic view. This impedes correlating system behavior across services, infrastructure, and network layers for root cause analysis. The diversity of deployment environments, ranging from more powerful fog nodes to much more constrained edge devices, still poses portability and scalability challenges. Many solutions include certain domain-specific assumptions, which limit generalizability with insufficiently addressed issues like resource overhead, mobility awareness, and diversified telemetry integration.

From a design perspective, the proposed taxonomy offers important guidance for future edge monitoring systems by pinpointing neglected dimensions and the most promising combinations of techniques. More specifically, such a system integration strategy should represent future research priorities centered on integrating cross-layer observability with lightweight telemetry mechanisms to provide unified system/service/network visibility. The coupling of kernel instrumentation under a low footprint (e.g., eBPF-based techniques) with intelligent adaptive analytics is a promising candidate in this regard. In the same way, using mobility-aware monitoring and standardized telemetry pipelines can bring great improvements to how adaptable and scalable the system is in different dynamic environments. These insights indicate that less explored dimensions and promising combinations of techniques provide valuable design guidance for future edge monitoring systems.

Overall, the findings emphasize that effective edge observability requires a holistic approach that goes beyond isolated metrics, logs, and traces that can be correlated across multiple layers of the system stack. These findings emphasize that effective edge observability requires a holistic approach that goes beyond isolated metrics, logs, and traces to enable their correlation across multiple layers of the system stack. This work serves to inform design priorities based on relevant gaps identified in existing research to accomplish robust, scalable, and intelligent observability solutions in the future on the evolving edge computing landscape.

Acknowledgement: The authors would like to express their sincere appreciation to the FAST School of Computing, National University of Computer and Emerging Sciences, Karachi, 75030, Pakistan, for the institutional support and resources that enabled this research.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Hamza Ahmed: Contributed to conceptualization, literature review, and initial manuscript drafting. Hassan Jamil Syed: Involved in designed the methodology, performed data analysis, reviewed and edited the manuscript, and supervised the project. Aqsa Aslam: Provide the key supervison and project supervision, help in study validation and provide the final manuscript revision. Sehar Zehra: Assisted in software review, figure and table preparation. Ummay Faseeha: Provided additional support in data verification, literature review, and manuscript formatting. Nurzati Iwani Othman: Contributed to conceptual guidance, provided critical insights into the research direction, and supported manuscript review and refinement to enhance the overall quality of the study. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jararweh Y, Doulat A, AlQudah O, Ahmed E, Al-Ayyoub M, Benkhelifa E. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In: Proceedings of the 2016 23rd International Conference on Telecommunications (ICT); 2016 May 16–18; Thessaloniki, Greece. doi:10.1109/ict.2016.7500486.
2. Choudhury A, Sarma KK, Gulvanskii V, Kaplun D, Dutta L. Leveraging federated learning and edge computing for pandemic-resilient healthcare. Sci Rep. 2025;15(1):20497. doi:10.1038/s41598-025-00199-9.
3. Al-Doghman F, Moustafa N, Khalil I, Sohrabi N, Tari Z, Zomaya AY. AI-enabled secure microservices in edge computing: opportunities and challenges. IEEE Trans Serv Comput. 2023;16(2):1485–504. doi:10.1109/tsc.2022.3155447.
4. Cao X. Dynamic management network system of automobile detection applying edge computing. Int J Netw Manag. 2023;33(5):e2231. doi:10.1002/nem.2231.
5. Syafrudin M, Fitriyani NL, Alfian G, Rhee J. An affordable fast early warning system for edge computing in assembly line. Appl Sci. 2019;9(1):84. doi:10.3390/app9010084.
6. Stojmenovic I, Wen S. The fog computing paradigm: scenarios and security issues. Proc 2014 Fed Conf Comput Sci Inf Syst. 2014;2:1–8. doi:10.15439/2014f503.
7. Zissis D, Lekkas D. Addressing cloud computing security issues. Future Gener Comput Syst. 2012;28(3):583–92. doi:10.1016/j.future.2010.12.006.
8. Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY. Edge intelligence: the confluence of edge computing and artificial intelligence. IEEE Internet Things J. 2020;7(8):7457–69. doi:10.1109/jiot.2020.2984887.
9. Raza S, Wang S, Ahmed M, Anwar MR. A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. Wirel Commun Mob Comput. 2019;2019(1):3159762. doi:10.1155/2019/3159762.
10. Bulkan U, Dagiuklas T, Iqbal M, Huq KMS, Al-Dulaimi A, Rodriguez J. On the load balancing of edge computing resources for on-line video delivery. IEEE Access. 2018;6:73916–27. doi:10.1109/access.2018.2883319.
11. Kyryk M, Pleskanka N, Pleskanka M, Nykonchuk P. Load balancing method in edge computing. In: Proceedings of the 2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET); 2020 Feb 25–29; Lviv-Slavske, Ukraine. p. 978–81.
12. Wang T, Zhang G, Liu A, Bhuiyan MZA, Jin Q. A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing. IEEE Internet Things J. 2019;6(3):4831–43. doi:10.1109/jiot.2018.2870288.

13. Liu G, Shi H, Kiani A, Khreishah A, Lee J, Ansari N, et al. Smart traffic monitoring system using computer vision and edge computing. IEEE Trans Intell Transp Syst. 2022;23(8):12027–38. doi:10.1109/tits.2021.3109481.
14. Wang T, Bhuiyan MZA, Wang G, Rahman MA, Wu J, Cao J. Big data reduction for a smart city's critical infrastructural health monitoring. IEEE Commun Mag. 2018;56(3):128–33. doi:10.1109/mcom.2018.1700303.
15. Dave R, Seliya N, Siddiqui N. The benefits of edge computing in healthcare, smart cities, and IoT. arXiv:2112.01250. 2021.
16. Hassan N, Yau KA, Wu C. Edge computing in 5G: a review. IEEE Access. 2019;7:127276–89. doi:10.1109/access.2019.2938534.
17. Liu Y, Peng M, Shou G, Chen Y, Chen S. Toward edge intelligence: multiaccess edge computing for 5G and Internet of Things. IEEE Internet Things J. 2020;7(8):6722–47. doi:10.1109/jiot.2020.3004500.
18. Soldani D, Nahi P, Bour H, Jafarizadeh S, Soliman MF, Di Giovanna L, et al. eBPF: a new approach to cloud-native observability, networking and security for current (5G) and future mobile networks (6G and beyond). IEEE Access. 2023;11:57174–202. doi:10.1109/access.2023.3281480.
19. Yu Y. Mobile edge computing towards 5G: vision, recent progress, and open challenges. China Commun. 2016;13(Suppl 2):89–99. doi:10.1109/cc.2016.7833463.
20. Chen B, Wan J, Celesti A, Li D, Abbas H, Zhang Q. Edge computing in IoT-based manufacturing. IEEE Commun Mag. 2018;56(9):103–9. doi:10.1109/mcom.2018.1701231.
21. Hassan N, Gillani S, Ahmed E, Yaqoob I, Imran M. The role of edge computing in Internet of Things. IEEE Commun Mag. 2018;56(11):110–5. doi:10.1109/mcom.2018.1700906.
22. Chen G, Lin W, Shi F, Zhang H, Wang B. PAWSSP: a two-stage parallelism-aware algorithm for joint workflow scheduling and service placement in edge computing. IEEE Trans Serv Comput. 2026;19(1):572–88. doi:10.1109/tsc.2025.3643326.
23. Cao K, Liu Y, Meng G, Sun Q. An overview on edge computing research. IEEE Access. 2020;8:85714–28. doi:10.1109/access.2020.2991734.
24. Cao J, Zhang Q, Shi W. Challenges and opportunities in edge computing. In: Edge computing: a primer. Cham, Switzerland: Springer International Publishing; 2018. p. 59–70. doi:10.1007/978-3-030-02083-5_5.
25. Großmann M, Schenk C. A comparison of monitoring approaches for virtualized services at the network edge. In: Proceedings of the 2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC); 2018 Dec 20–21; Hamammet, Tunisia. p. 85–90.
26. Kong X, Wu Y, Wang H, Xia F. Edge computing for Internet of everything: a survey. IEEE Internet Things J. 2022;9(23):23472–85. doi:10.1109/jiot.2022.3200431.
27. Pydi H, Iyer GN. Analytical review and study on load balancing in edge computing platform. In: Proceedings of the 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC); 2020 Mar 11–13; Erode, India. p. 180–87.
28. Zehra S, Syed HJ, Samad F, Faseeha U. DeSFAM: an adaptive eBPF and AI-driven framework for securing cloud containers in real time. IEEE Access. 2025;13:139203–24. doi:10.1109/access.2025.3592192.
29. Zehra S, Syed HJ, Samad F, Faseeha U, Ahmed H, Khurram Khan M. Securing the shared kernel: exploring kernel isolation and emerging challenges in modern cloud computing. IEEE Access. 2024;12:179281–317. doi:10.1109/access.2024.3507215.
30. Wu H, Lin W, Zhang H, Shi F, Shen W, Li K, et al. Container scheduling strategy based on image layer reuse and sequential arrangement in mobile edge computing. IEEE Trans Mobile Comput. 2025;24(9):8700–13. doi:10.1109/tmc.2025.3557160.
31. Satyanarayanan M. The emergence of edge computing. Computer. 2017;50(1):30–9. doi:10.1109/mc.2017.9.
32. Shi W, Cao J, Zhang Q, Li Y, Xu L. Edge computing: vision and challenges. IEEE Internet Things J. 2016;3(5):637–46. doi:10.1109/jiot.2016.2579198.
33. Gowtham V, Keil O, Yeole A, Schreiner F, Tschoke S, Willner A. Determining edge node real-time capabilities. In: Proceedings of the 2021 IEEE/ACM 25th International Symposium on Distributed Simulation and Real Time Applications (DS-RT); 2021 Sep 27–29; Valencia, Spain. doi:10.1109/ds-rt52167.2021.9576132.

34. Shahinfar F, Miano S, Panda A, Antichi G. Demystifying performance of eBPF network applications. Proc ACM Netw. 2025;3:1–21. doi:10.1145/3749216.
35. Tang X, Long B, Zhou L. Real-time monitoring and analysis of track and field athletes based on edge computing and deep reinforcement learning algorithm. Alex Eng J. 2025;114(7):136–46. doi:10.1016/j.aej.2024.11.024.
36. Mr krishnamoorthy J, Egneswari K. Mobility and dependence-aware QoS monitoring in mobile edge computing. J Adv Zool. 2023;44(3):386–91. doi:10.17762/jaz.v44i3.653.
37. Wei B, Xie Z, Liu Y, Wen K, Deng F, Zhang P. Online monitoring method for insulator self-explosion based on edge computing and deep learning. CSEE J Power Energy Syst. 2021;8(6):1684–96. doi:10.17775/cseejpes.2020.05910.
38. Fournier G, Afchain S, Baubeau S. Runtime security monitoring with eBPF. 2021 [cited 2026 Jan 1]. Available from: https://www.sstic.org/media/SSTIC2021/SSTIC-actes/runtime_security_with_ebpf/SSTIC2021-Article-runtime_security_with_ebpf-fournier_afchain_baubeau.pdf.
39. Otero M, García JM, Fernandez P. An extensible lightweight framework for distributed telemetry of microservices. Sustain Comput Inform Syst. 2025;46(5):101100. doi:10.1016/j.suscom.2025.101100.
40. Morabito G, Ficara A, Celesti A, Villari M, Fazio M. Consensus-based distributed orchestration framework for microservices in edge computing clusters. Future Gener Comput Syst. 2026;176(1):108221. doi:10.1016/j.future.2025.108221.
41. Huang Y, Lu Y, Wang F, Fan X, Liu J, Leung VCM. An edge computing framework for real-time monitoring in smart grid. In: Proceedings of the 2018 IEEE International Conference on Industrial Internet (ICII); 2018 Oct 21–23; Seattle, WA, USA. doi:10.1109/icii.2018.00019.
42. Zhang P, Zhang Y, Dong H, Jin H. Mobility and dependence-aware QoS monitoring in mobile edge computing. IEEE Trans Cloud Comput. 2021;9(3):1143–57. doi:10.1109/tcc.2021.3063050.
43. Arroba P, Buyya R, Cárdenas R, Risco-Martín JL, Moya JM. Sustainable edge computing: challenges and future directions. Softw Pract Exp. 2024;54(11):2272–96. doi:10.1002/spe.3340.
44. Samanta A, Esposito F, Nguyen TG. Fault-tolerant mechanism for edge-based IoT networks with demand uncertainty. IEEE Internet Things J. 2021;8(23):16963–71. doi:10.1109/jiot.2021.3075681.
45. Chen J, Mao F, Lv Z, Tang J. EdgeFD: an edge-friendly drift-aware fault diagnosis system for industrial IoT. In: Proceedings of the 2023 IEEE 23rd International Conference on Communication Technology (ICCT); 2023 Oct 20–22; Wuxi, China. p. 390–6. doi:10.1109/icct59356.2023.10419797.
46. Wang R, Lai J, Zhang Z, Li X, Vijayakumar P, Karuppiah M. Privacy-preserving federated learning for Internet of medical things under edge computing. IEEE J Biomed Health Inform. 2023;27(2):854–65. doi:10.1109/jbhi.2022.3157725.
47. Nouri A, Bozga M, Molnos A, Legay A, Bensalem S. *ASTROLABE*: a rigorous approach for system-level performance modeling and analysis. ACM Trans Embed Comput Syst. 2016;15(2):1–26. doi:10.1145/2885498.
48. Van Renesse R, Birman KP, Vogels W. Astrolabe: a robust and scalable technology for distributed system monitoring, management, and data mining. ACM Trans Comput Syst. 2003;21(2):164–206. doi:10.1145/762483.762485.
49. Yalagandula P, Dahlin M. A scalable distributed information management system. SIGCOMM Comput Commun Rev. 2004;34(4):379–90. doi:10.1145/1030194.1015509.
50. Brandón Á, Pérez MS, Montes J, Sanchez A. FMonE: a flexible monitoring solution at the edge. Wirel Commun Mob Comput. 2018;2018(1):2068278. doi:10.1155/2018/2068278.
51. Rac S, Brorsson M. Cost-aware service placement and scheduling in the edge-cloud continuum. ACM Trans Archit Code Optim. 2024;21(2):1–24. doi:10.1145/3640823.
52. Taherizadeh S, Taylor I, Jones A, Zhao Z, Stankovski V. A network edge monitoring approach for real-time data streaming applications. In: Economics of grids, clouds, systems, and services. Cham, Switzerland: Springer International Publishing; 2017. p. 293–303. doi:10.1007/978-3-319-61920-0_21.
53. Ahmed H, Syed HJ, Sadiq A, Ibrahim AO, Alohaly M, Elsadig M. Exploring performance degradation in virtual machines sharing a cloud server. Appl Sci. 2023;13(16):9224. doi:10.3390/app13169224.
54. Syed HJ, Gani A, Nasaruddin FH, Naveed A, Ahmed AIA, Khurram Khan M. CloudProcMon: a non-intrusive cloud monitoring framework. IEEE Access. 2018;6:44591–606. doi:10.1109/access.2018.2864573.

55. Iqbal S, Khan TM, Naqvi SS, Naveed A, Usman M, Khan HA, et al. LDMRes-Net: a lightweight neural network for efficient medical image segmentation on IoT and edge devices. IEEE J Biomed Health Inform. 2023;28(7):3860–71. doi:10.1109/jbhi.2023.3331278.
56. Korenivska OL, Benedytskyi VB, Andreiev OV, Medvediev MG. A system for monitoring the microclimate parameters of premises based on the Internet of Things and edge devices. J Edge Comp. 2023;2(2):125–47. doi:10.55056/jec.614.
57. Lu S, Lu J, An K, Wang X, He Q. Edge computing on IoT for machine signal processing and fault diagnosis: a review. IEEE Internet Things J. 2023;10(13):11093–116. doi:10.1109/jiot.2023.3239944.
58. Rajagopal SM, Supriya M, Buyya R. FedSDM: federated learning based smart decision making module for ECG data in IoT integrated Edge–Fog–Cloud computing environments. Internet Things. 2023;22(3):100784. doi:10.1016/j.iot.2023.100784.
59. Taherizadeh S, Jones AC, Taylor I, Zhao Z, Stankovski V. Monitoring self-adaptive applications within edge computing frameworks: a state-of-the-art review. J Syst Softw. 2018;136(9):19–38. doi:10.1016/j.jss.2017.10.033.
60. Habib ur Rehman M, Jayaraman P, Malik S, Khan A, Medhat Gaber M. RedEdge: a novel architecture for big data processing in mobile edge computing environments. J Sens Actuator Netw. 2017;6(3):17. doi:10.3390/jsan6030017.
61. Mehmood H, Khalid A, Kostakos P, Gilman E, Pirttikangas S. A novel Edge architecture and solution for detecting concept drift in smart environments. Future Gener Comput Syst. 2024;150(5):127–43. doi:10.1016/j.future.2023.08.023.
62. Wang T, Zhang G, Bhuiyan MZA, Liu A, Jia W, Xie M. A novel trust mechanism based on fog computing in sensor–cloud system. Future Gener Comput Syst. 2020;109(6):573–82. doi:10.1016/j.future.2018.05.049.
63. Zhang DG, Ni CH, Zhang J, Zhang T, Yang P, Wang JX, et al. A novel edge computing architecture based on adaptive stratified sampling. Comput Commun. 2022;183(5):121–35. doi:10.1016/j.comcom.2021.11.012.
64. Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the internet of things. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing; 2012 Aug 17; Helsinki, Finland. doi:10.1145/2342509.2342513.
65. Faseeha U, Jamil Syed H, Samad F, Zehra S, Ahmed H. Observability in microservices: an in-depth exploration of frameworks, challenges, and deployment paradigms. IEEE Access. 2025;13(1):72011–39. doi:10.1109/access.2025.3562125.
66. Abderrahim M, Ouzzif M, Guillouard K, Francois J, Lebre A. A holistic monitoring service for fog/edge infrastructures: a foresight study. In: Proceedings of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud); 2017 Aug 21–23; Prague, Czech Republic. p. 337–44.
67. Gaddam MK. Architecting observability for AI-driven microservices at scale. In: 2025 3rd International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI); 2025 Sep 17–19; Coimbatore, India. doi:10.1109/icoici65217.2025.11252857.
68. Nimmagadda S. Applying AI/ML to Kubernetes logging and monitoring in enhancing observability through intelligent systems. Eur J Comput Sci Inf Technol. 2025;13(49):141–52. doi:10.37745/ejcsit.2013/vol13n49141152.
69. Hudson N, Khamfroush H, Baughman M, Lucani DE, Chard K, Foster I. QoS-aware edge AI placement and scheduling with multiple implementations in FaaS-based edge computing. Future Gener Comput Syst. 2024;157(8):250–63. doi:10.1016/j.future.2024.03.035.
70. Xie Y, Wu Q, Fan P, Cheng N, Chen W, Wang J, et al. Resource allocation for twin maintenance and computing task processing in digital twin vehicular edge computing network. arXiv:2407.07575. 2024.