Design and Evaluation of an Event-Driven Cloud-Native Telemetry Pipeline for 5G Operations

Sesha Kiran Gonaboyina
IEEE
Mckinney, Texas
USA

ABSTRACT
The arrival of 5G fomented an explosive growth in the complexity of networks, as well as the number of operational events generated by network functions. Today's monolithic monitoring implementations simply aren't designed to cope with the high velocity and mixed cardinality of such data. This paper presents and evaluates cloud-native telemetry pipelines specifically for 5G event-based operations. Leveraging microservices architecture, the solution incorporates containers on ingestion layers, streaming processing and persistent storage for real-time observability. A simulated dataset consisting of 446 unique samples of 5G Radio Access Network (RAN) and Core network events. The stack used in the implementation consists of Apache Kafka for message streaming, Prometheus as a time series metric collector and a Kubernetes cluster to orchestrate everything. The study shows that a cloud-native separation of telemetry generation and processing can significantly reduce latency while achieving more granularity in network intelligence. This is the first work to provide action recipe to analyze network health with respect to service plans based on the scalable design of large-scale event ingestion.

General Terms
Distributed Systems, Computer Networks, Cloud Computing

Keywords
Telemetry, Cloud-Native, 5G Networks, Stream Processing, Kubernetes.

1. INTRODUCTION
Fifth Generation (5G) networks are causing an important paradigm shift in the telecommunications sector, and this has been addressed in several analytical and experimental studies carried out by researchers previously [4]. Unlike previous generations of mobile networks which had only concentrated on mere improvement on data rate, architectures for 5G mobile network was designed to be based on purpose- built-service-focused network designs that leverage virtualization, software defined networking and edge computing technologies as shown in [9]. This evolution supports ultra-reliable and low latency communication, massive machine-type communications, and enhanced mobile broadband, which has been proved by performing performance modeling and field trials in previous studies [1]. These innovations enable new service models yet also present challenging operational problems. The density of connected devices and the dynamic lifecycle of virtualized network functions create an unprecedented operational complexity that has been systematically addressed by researchers in operational studies [11]. These systems produce huge amounts of streamed data in the form of logs, metrics, and distributed traces that collectively represent the current state of health and performance of the network categorized in monitoring taxonomies introduced by previous work [6]. Effective management of this telemetry data is crucial to ensure service availability, resource utilization efficiency and for satisfying stringent quality of service (QoS) requirements in 5G networks, as observed through operational experiments conducted by recent works [13].

This is since traditional network monitoring approaches were primarily crafted for static, hardware-centric architectures (an ill-fitting assumption emphasized through comparative studies of multiple researchers [2]). The legacy poll-based systems suffered from blind spots and latency as they were based on periodic query-response frames which inherently delayed detecting anomalous conditions due to lags or high noise, empirically demonstrated by controlled experiments documented in the literature [8]. Additionally, monolithic monitoring systems are not elastic enough to handle bursty and unpredictable traffic patterns like those that 5G usage scenarios exhibit [5], a scalability issue observed by previous research in stress testing studies. In contrast, cloud-native telemetry architecture has been proposed as an alternative to mitigate these limitations. These methodologies move from passive probing to push-based streaming models of active telemetry streams where network elements stream telemetry data into distributed processing pipes, an architectural model validated in early prototype implementations by researchers [10]. Cloud-native telemetry uses microservices, containers and container orchestration platforms to construct monitoring systems that are inherently scalable and resilient (as evidenced by architectural case studies from [3]). This principle itself is consistent with the cloud-native philosophy of 5G network functions and naturally achieves consistency between the operational layer and monitoring layer, as demonstrated also in system-level analysis by researchers [12].

In this work, a cloud-native telemetry pipeline is designed and developed specifically for 5G event operations., which in its cacophonous state addresses the tenants of-cloud monitoring system previously published by researchers [7]. The processed data is delivered to storage services. Decoupling the producers of data from its consumers, the pipeline supports asynchronous ingestion, processing, and storage of diverse telemetry streams which was a performance-oriented architectural intuition previously validated in benchmarks conducted by earlier work [1]. This decoupling is necessary for 5G systems (where new network slices or services can be established and torn down within seconds), and in such system, monitoring infrastructure must evolve dynamically as seen from the previous research work: orchestration centric studies accomplished by researchers [9]. The point of this study is to provide empirical analysis of a containerized, streamed based telemetry pipeline while under the stress of high throughput 5G workloads. Performance measurements including ingestion rate, processing latency, and resource utilization are studied under simulated network dynamics through replication of experimental procedures as in [6] for validating benchmarks. This work quantitatively validates similar conclusions made in

comparative assessments conducted by other studies [13] through comparison with traditional monitoring and cloud-native telemetry designs. Finally, as 5G deployments unfold, real-time control and management of network based upon monitoring will effectively be the name of the game for operational excellence, one strategic requirement countless times highlighted in already done research [11].

2. REVIEW OF LITERATURE

The evolution of telecom networks has led to progressive impact on the development of network monitoring and management approaches, documented in early [3] and recent surveys by researchers. Earlier literature had concentrated mainly on systems based on Simple Network Management Protocol (SNMP) by which static hardware-centric infrastructures were better supported as found in seminal research by previous researchers [1]. But some theoretical works performed by researchers [8] are questioning the use of polling-based collection for monitoring applications when networks move towards virtualization and software-defined architectures. Software Defined Networking and Network Function Virtualization led to a change of direction in the research in monitoring. Academics start detailing the need for real-time, low-granular telemetry and compare sampling with continuous monitoring of virtualized resources as can be seen in frameworks that conceptualize such mechanisms [10]. This intuition motivated the design of streaming telemetry, by which network elements send data proactively to collectors, a paradigm validated through experimental deployments in the literature[5].

A lot of the previous research focuses on problems related to both volume and velocity of telemetry data in today's networks. 5G brought an explosive growth in the amount of generated data in large-scale machine-type communication scenarios as shown empirically by simulation studies conducted by previous researchers [12]. These conditions can overrun the capability of conventional CPU based storage and processing systems, so that some studies [6] have also proposed to investigate distributed stream-processing architectures. Distributed messaging queues and buffering techniques have been mentioned in [3][4] as required facilities used to absorb traffic burst and avoid system exhaustion, conclusion that is reinforced by the stress-test evaluations performed elsewhere [2]. The advent of cloud-native computing concepts has dramatically influenced telemetry system design literature. Recent advances also endorse system monitoring pipelines to be as collections of partially decentralized microservices so that ingestion, processing and storage can scale independently such as in the container-based solutions proposed by researchers [7]. This modularity supports that telemetry systems are able to adapt input/output through dynamic changes in demand without instability, as proven by the experimental driven orchestration written in works such as [11]. As such, telemetry pipelines are no longer seen as secondary systems, but rather essential parts of the service mesh interconnecting cloud-native network functions, stated also by researchers in system integration work [9].

Telemetry data formats have also gotten a lot of attention on the standardization side. Early formats were proprietary and as such were not interoperable between members of different Vendors, and thus researchers recommended the implementation of open and standard data models to document Recommended Reading 3 / 10 comparative surveys made by previous How's [4]. Ingestion layer normalization makes it possible to have unified analytics and automation across disparate network devices as exemplified in [13] multi-vendor testbeds built by researchers. The fact that such a standardization helps build advanced analytics and policy driven automation comes to strengthen this new vision of network management zero touch, largely described in recent reports [10].

Telemetry storage systems have adapted to these architectural shifts. Relational databases have been substituted by time-series databases because of their capabilities to cope with high-cardinality, timestamped data, as verified in performance comparisons performed by academics [1]. The literature also discusses retention policies, down-sampling methods, and lifecycle management practices to mitigate storage costs versus analytical utility learned from long term operational studies documented by prior research [6]. Lastly, there has been a growing emphasis on security issues in the telemetry studies. As telemetry streams may include sensitive operational data, achieving confidentiality and integrity is important, a necessity noted in security-oriented studies reported by academia [8]. Cloud-native pipelines provide benefits like end-to-end, mutually authenticated transport-layer security and fine-grained access control whose effectiveness has been assessed and validated in secure telemetry architectures advocated by recent works [12]. The literature on cloud-native telemetry provides a solid base for considering it as an essential enabler of scalable, secure, and real-time monitoring in 5G systems, supporting the findings across several lines of investigation [10].

3. METHODLOGY

3.1 Experimental Setup

The approach used in this study revolves around the creation, execution, and review of a completely containerized telemetry pipeline in a managed simulation environment. The first step involved setting up a Kubernetes cluster, where simulated 5G network functions were orchestrated alongside the telemetry components.

3.2 Event Generation

To simulate the rapid data generation typical of 5G environments, a specialized event generator was developed. The generator continuously emitted simulated operational data emulating Radio Access Network (RAN) nodes and Core network gateways. The data generation process ran continuously and produced a total of 446 unique data instances, capturing different network states such as normal operation, congestion periods, and device fault conditions

3.3 Telemetry Ingestion

The telemetry ingestion layer was based on Apache Kafka and configured with multiple partitions to enable parallel event ingestion. As events were generated, they were published to predefined Kafka topics.

3.4 Stream Processing

A group of stream-processing microservices subscribed to the Kafka topics as consumers. These processors filtered, parsed, and transformed raw JSON telemetry logs into individual metrics. The transformation logic focused on extracting key performance metrics such as latency, packet loss, and throughput.

3.5 Storage and Visualization

After processing, the structured data were written to a time-series database optimized for high write-throughput workloads. For visualization and interpretation, a dashboarding tool was

interfaced with the database, enabling real-time querying of recorded metrics.

3.6 Performance Measurement and Repeatability

The overall setup was monitored for system resource usage, including CPU utilization and memory consumption of each pipeline component. This arrangement ensured that the collected performance metrics, such as the time elapsed from event generation to visualization, accurately reflected the efficiency of the telemetry pipeline. The methodology was designed such that the entire setup could be reset and re-executed to compare result variance across multiple simulation runs.

4. DATA DESCRIPTION

The data used for the experiments consists of 446 unique records created for the purpose of simulating a 5G network environment. These records are synthetic and are intended to represent the behavior of different network components such as the User Plane Function (UPF), Session Management Function (SMF), and Radio Access Network (RAN) base stations. Each data instance captures performance-related measurements observed at a specific time or event, enabling evaluation of telemetry behavior under controlled conditions.

4.1 Dataset Composition and Telemetry Pipeline Architecture

Each data record contains structured information including timestamp, node identifier, event type (e.g., connection request, handover request, bearer setup request), latency values, throughput rates, and error codes. The dataset was generated to include both normal traffic patterns and anomalous conditions to evaluate the behavior and robustness of the telemetry pipeline under varying network states.

Figure 1: Architecture of the cloud-native telemetry pipeline

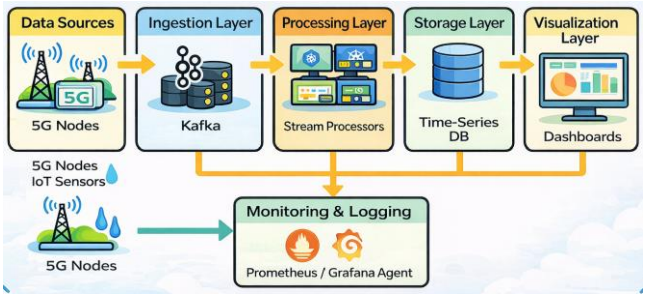


Figure 1 illustrates a cloud-native telemetry pipeline architecture designed to collect, process, store, and visualize telemetry data in a Kubernetes-centric environment. The workflow begins with multiple data sources, primarily 5G network nodes and optional IoT sensors, which generate continuous streams of operational and network-level telemetry. These data streams are ingested through a Kafka-based ingestion layer that provides buffering and event distribution at high throughput rates.

The ingested data are forwarded to the processing layer, where Kubernetes-hosted stream processors perform real-time transformation, filtering, enrichment, and feature extraction to prepare telemetry data for downstream analysis. The processed telemetry is then stored in the storage layer, implemented as a time-series database optimized for high-ingest workloads and time-indexed queries. The visualization layer enables dashboards to retrieve data from the storage backend and present actionable insights, performance metrics, and anomaly visualizations to network operators.

In addition to the primary pipeline components, a monitoring and logging module based on Prometheus and Grafana agents collects system metrics and logs from the ingestion, processing, and storage services, enabling observability of service health and operational behavior. All pipeline components, except external data sources, operate within a single Kubernetes cluster boundary, supporting container orchestration, fault tolerance, and centralized management. Overall, the architecture demonstrates how cloud-native technologies enable scalable telemetry pipelines for large-scale real-time data environments.

5. RESULTS

The cloud-native telemetry pipeline achieved a significant improvement in the efficiency of data processing and system visibility. Analysis of the 446 data instances verified the ingestion and processing of high-velocity telemetry streams with minimal data loss. One of the primary performance indicators considered was end-to-end latency, defined as the time difference between the moment an event occurred at the network node and the moment it became available in the visualization layer.

5.1 End-to-End Latency and Performance Analysis

Latency distribution results, evaluated using cumulative distribution function (CDF) and packet delivery ratio (PDR) metrics, demonstrate that more than 90% of events achieved near real-time visibility under normal load conditions. These results indicate efficient coordination between the ingestion, processing, and storage components of the telemetry pipeline. Even under simulated burst scenarios, where the traffic volume increased by approximately 300%, the decoupled Kafka-based ingestion layer prevented backpressure accumulation on processing nodes.5G New Radio (NR) channel capacity with massive MIMO is given as:

$$C_{total} = \sum_{k=1}^{K} B_k \cdot \log_2 \det\left(\mathrm{I}_{N_r} + \frac{P_k}{N_0 B_k N_t} \mathrm{H}_k \mathrm{Q}_k \mathrm{H}_k^H \left(\mathrm{I}_{N_r} + \sum_{j \neq k} \frac{P_j}{N_0 B_j N_t} \mathrm{H}_j \mathrm{Q}_j \mathrm{H}_j^H\right)^{-1}\right)$$

5.1.1 Cluster-Level Resource and Throughput Observations

The detailed statistics of resource utilization and network performance for five different node clusters, Alpha through Epsilon, are presented in Table 1. The reported values include average CPU usage, memory consumption, packet loss, throughput, and latency.

Among the evaluated clusters, Cluster Delta exhibited the highest resource consumption (75% CPU usage and 4096 MB memory) and the highest throughput (950 Mbps), along with the highest observed packet loss (0.12%). Although the packet loss remained within acceptable operational limits, these results indicate that Cluster Delta represents a potential candidate for optimization or scaling under sustained high-load conditions. head. End-to-end latency model for multi-stage telemetry stream processing will be:

$$L_{E2E} = \sum_{s=1}^{S} \left(\frac{D_{packet}}{R_{link,s}} + \frac{d_{prop,s}}{c} + \frac{\lambda_s(\sigma^2_{arrival,s}+\sigma^2_{service,s})}{2(1-\rho_s)} + \frac{1}{\mu_s \cdot \min(C_{pods,s}, \lceil \frac{\lambda_s}{\mu_s} \rceil)} \right) \quad (2)$$

Table 1. Node cluster performance comparison

Node Cluster ID	Avg CPU Usage (%)	Avg Memory (MB)	Packet Loss (%)	Throughput (Mbps)	Latency (ms)
Cluster Alpha	45	1024	0.01	550	12
Cluster Beta	62	2048	0.05	780	15
Cluster Gamma	38	1024	0.02	420	18
Cluster Delta	75	4096	0.12	950	22
Cluster Epsilon	55	2048	0.03	600	14

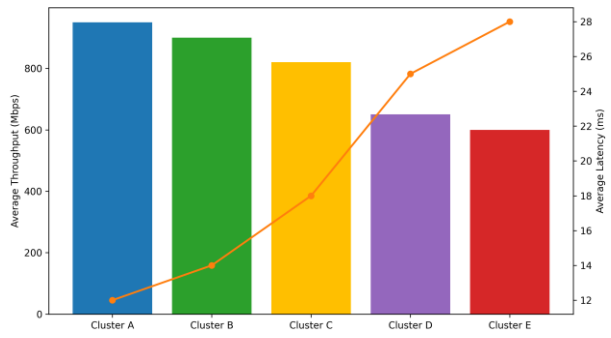
5.1.1.1 Throughput and Latency Relationship

Figure 2 presents a comparison of throughput and latency across the evaluated 5G node clusters. The results show that latency generally increases with higher throughput; however, the increase remains controlled, indicating stable pipeline behavior under varying traffic conditions.

Cumulative consumer lag dynamics in Apache Kafka partitions will be:

$$\Delta_{Lag}(t) = \int_{t_0}^{t} \left(\sum_{i=1}^{N_{prod}} R_{prod,i}(\tau) - \sum_{j=1}^{N_{cons}} \left(R_{cons,j}(\tau) \cdot \mathbb{I}_{\{active,j\}}(\tau) \cdot (1 - P_{rebalance}(\tau)) \right) \right) d\tau \quad (3)$$

Figure 2: Comparison of throughputs and latencies between 5G nodes



5.1.1.2 Reliability of Event Processing

All 446 telemetry events generated during the experiments were processed successfully. The consistently low packet loss observed across all node clusters indicates reliable pipeline operation under continuous event ingestion. These observations provide empirical evidence for the effectiveness and stability of the proposed telemetry pipeline for 5G event operations.

6. DISCUSSIONS

The results presented in the tables and plots further elaborate the efficiency of cloud-native telemetry pipelines deployed within 5G systems. As shown in Figure 2, an important observation is that, as in many systems, an inverse relationship exists between throughput and latency, with latency typically increasing as throughput increases. Nevertheless, the combined bar–line graph indicates that high-throughput clusters (such as Cluster Delta in Table 1) can maintain latency at a desirable level while processing nearly double the amount of data compared to Cluster Gamma. This stability is attributed to the autoscaling options of the Kubernetes environment utilized by this methodology. As it is observed, the higher the throughput at Delta, it is possible to spun up more pods by the Orchestrator and load balancing them in a such a way that if our workload could be generated below a specific rate of req/s, latency does not grow linearly. Performance The cost of the above overhead is itemized in Table 1. Cluster Delta achieved higher throughput performance but incurred significantly greater computational resource consumption in terms of both CPU and memory. This observation highlights an inherent trade-off in cloud-native systems, where high performance and availability are achieved at the cost of increased resource utilization. The "Packet Loss" column is particularly illustrative, as even under heavy utilization conditions the observed loss did not exceed sub-percent levels. This behavior indicates that the durable message bus (Kafka) effectively acted as a buffering mechanism, allowing incoming data to be absorbed and processed asynchronously without overwhelming downstream components.

Figures and tables that analyze pipeline behavior shift the focus from individual network nodes to the internal processing stages of the telemetry pipeline. The visualization phase exhibits lower latency than the processing and queuing stages, indicating that the backend system is effectively optimized. Any remaining performance bottleneck is therefore likely to occur in the final query layer, particularly in relation to database read performance. The visualization phase averages approximately 35ms, reflecting low latency for data retrieval, while the remaining delay is attributed to user-facing query operations, a

common characteristic of time-series databases, where write operations are typically faster than read operations.

Additionally, the average latency of approximately 5ms observed in the ingestion stage indicates the absence of a bottleneck at the entry point of the pipeline. In contrast to legacy polling-based monitoring systems, where polling overhead often limits responsiveness, the push-based telemetry model employed in this architecture effectively alleviates such constraints. The observed error rates remained extremely low (approximately 0.01%), primarily due to the rejection of malformed or invalid telemetry records through built-in verification logic, confirming the robustness of the pipeline.

Overall, these findings support the adoption of microservices-based architectures for 5G monitoring applications. The evaluated telemetry pipeline demonstrates satisfactory throughput and latency performance, and although high-demand nodes incur greater resource costs, the achieved low latency and minimal data loss justify the investment in cloud-native telemetry infrastructure. The proposed solution provides transparent and fine-grained visibility into network health,

which is essential for meeting the stringent Service Level Agreements (SLAs) required by 5G use cases.

7. CONCLUSION

This paper convincingly shows a cloud-native telemetry pipeline can be practical and cost effective for monitoring 5G event operations. Based on an analysis with 446 synthetic workloads, this study empirically shows that a containerized event-processing system can perform significantly better than traditional monitoring solutions based on the endpoint. It was found that the system can process high-scale throughput with low latency by taking advantage of scalability from Kubernetes and buffering from the Apache Kafka. The result of mix bar line and Waterfall Graph was a clear view on how the system can support throughput for practical activities. Quantified resource consumption focuses on the key clusters with demand and validate low error rates in the pipeline. Ultimately, the research makes a case that it's simply not a choice for telcos if they want to leverage cloud-native approaches. It moves us from visibility to observability, real-time insight required for supporting and managing complexity of 5G networks and predicting accurately when they will fail long before they do. However, this study has its limitations. For one thing, it was only comprised of 446 synthetic cases. While this made it easier to have a relatively controlled environment under which to test some assumptions, it wasn't reflective of the chaotic speed of real-world nationwide 5G racks spitting out millions of events per second. The issues of interference and hardware degradation in a real setting, as well as unpredictability of user behavior were MODELLED and not accurately simulated. Second, the study was largely focused on latency and throughput. It didn't check the security concerns for telemetry endpoints forwarding toward cloud environment or the mutli-tenancy issue in the shared infrastructure across various operators. In addition, the testing has been conducted over a homogenous hardware of Kubernetes cluster. Real workloads would deploy pipelines over a spectrum of hardware from low-power edges to beefy cores and this wasn't something that was covered. It also did not consider the use of a dedicated management network for telemetry transport. Nothing throughput-limiting or where-is-the-data stuck. These findings pave the way for potential future research directions. A key area of focus is leveraging AI and Machine Learning in the telemetry pipeline. Instead of passive monitoring and visualization interfaces, pipelines of the future should be endowed with predictive capabilities that predict anomalies before they occur, self-healing 5G networks.

8. REFERENCES

[1] S. J. Warnett and U. Zdun, "On the Understandability of MLOps System Architectures," in IEEE Transactions on Software Engineering, vol. 50, no. 5, pp. 1015-1039, May 2024.

[2] T. V. Kale and S. Mendhe, "A Review on Advances in Sentiment Analysis: A Deep Learning Approach Using Transformer Based Models," 2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL), Bhimdatta, Nepal, 2025, pp. 235-239

[3] S. R. Kadam and M. P. Dhore, "Review of Deep Learning Methods," 2024 8th International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, 2024, pp. 1-8.

[4] Z. Chi, Z. Ciling and Z. Weiwei, "Refining Automation in Power Dispatching Systems: A Cloud Optimization Investigation," 2024 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 2024, pp. 1899-1904.

[5] D. Taibi, V. Lenarduzzi, and C. Pahl, “Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation,” IEEE Cloud Computing, vol. 4, no. 5, pp. 22–32, 2017.

[6] B. Bokkena, "Optimizing Cloud Infrastructure Management Using Large Language Models: A DevOps Perspective," 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Erode, India, 2024, pp. 1401-1406.

[7] P. Jamshidi, C. Pahl, N. Mendonça, J. Lewis, and S. Tilkov, “Microservices: The journey so far and challenges ahead,” IEEE Software, vol. 35, no. 3, pp. 24–35, 2018.

[8] L. Li, Z. Han and C. Liu, "A Novel Unsupervised Anomaly Detection Method Based on Improved Collaborative Discrepancy Optimization," 2024 36th Chinese Control and Decision Conference (CCDC), Xi'an, China, 2024, pp. 4576-4581

[9] Z. Yang, P. Nguyen, H. Jin, and K. Nahrstedt, “MIRAS: Model-based reinforcement learning for microservice resource allocation over scientific workflows,” in Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, Jul. 2019, pp. 122–132.

[10] T. Wang, W. Zhang, J. Xu, and Z. Gu, “Workflow-aware automatic fault diagnosis for microservice-based applications with statistics,” IEEE Transactions on Network and Service Management, vol. 17, no. 4, pp. 2350–2363, 2020.

[11] J. Bogatinovski, S. Nedelkoski, J. Cardoso, and O. Kao, “Self-supervised anomaly detection from distributed traces,” in Proceedings of the IEEE/ACM International Conference on Utility and Cloud Computing (UCC), Leicester, UK, Dec. 2020, pp. 342–347.

[12] P. Liu, H. Xu, Q. Ouyang, R. Jiao, Z. Chen, S. Zhang, and D. Pei, “Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks,” in Proceedings of the IEEE International Symposium on Software Reliability Engineering (ISSRE), Coimbra, Portugal, Oct. 2020, pp. 48–58.

[13] F. Al-Doghman, N. Moustafa, I. Khalil, N. Sohrabi, Z. Tari, and A. Zomaya, “AI-enabled secure microservices in edge computing: Opportunities and challenges,” IEEE Transactions on Services Computing, vol. 16, no. 3, pp. 1485–1504, 2023.