Securing and Sustaining IoT Edge-Computing Architectures through Nanoservice Integration

Cinthya Celina Tamayo Gonzalez, Ijaz Ahmad, *Graduate Student Member, IEEE*
Simone Soderi, *Senior Member, IEEE*, Erkki Harjula, *Member, IEEE*

***Abstract*—The rapid proliferation of the Internet of Things (IoT) and edge computing devices calls for solutions that deliver low latency, energy efficiency, and robust security—often challenging goals to balance simultaneously. This paper introduces a novel nanoservice-based framework that dynamically adapts to changing demands while achieving sustainable and secure edge operations. By breaking down functionalities into specialized and narrowly scoped nanoservices that are requested only as needed and eliminated when idle, the approach significantly reduces latency and energy usage compared to conventional, more static methods. Moreover, integrating a Zero-Trust Architecture (ZTA) ensures that every component—computational or security-related—is continuously verified and restricted through strict access controls and micro-segmentation. This framework's adaptability extends uniformly to all nanoservices, including those providing security features, thereby maintaining strong protective measures even as workloads and network conditions evolve. Experimental evaluations on IoT devices under varying workloads demonstrate that the proposed approach significantly reduces energy consumption and latency while maintaining security and scalability. These results underscore the potential for an integrated, flexible model that simultaneously addresses energy efficiency, performance, and security—an essential trifecta in future edge computing environments.**

***Index Terms*—Internet of Things (IoT), smart devices, energy, memory, resource constraints, edge computing, edge-cloud continuum.**

I. Introduction

Over the past two decades, the proliferation of IoT technologies has driven the widespread adoption of edge computing. With billions of interconnected devices deployed across diverse sectors such as healthcare, manufacturing, transportation, and smart cities, edge computing has emerged as a critical enabler of modern digital ecosystems. By positioning computational and data storage resources closer to where data is generated, *edge computing* reduces latency, enhances real-time processing capabilities, and conserves network bandwidth. Despite its transformative potential, edge computing also introduces unique security challenges. Addressing these challenges necessitates robust security frameworks that can adapt to edge computing' dynamic and distributed nature, ensuring data and services' confidentiality, integrity, and availability while preserving user privacy [1].

Zero-Trust Architecture (ZTA) rejects implicit trust based on location or ownership, a principle particularly relevant to edge computing in 6G contexts [2]. These networks not only bring computation closer to users, but also introduce novel security challenges due to their decentralized structure and reliance on modular service components, including microservices and their more lightweight variant, *nanoservices* [1]. Traditional security approaches fall short, necessitating the zero-trust model's continuous verification and stringent access control, irrespective of service location. In edge computing, ZTA ensures security through rigorous identity checks and the principle of least privilege, significantly reducing unauthorized access risks. It also employs micro-segmentation, dividing the network into smaller zones with controlled access, thus minimizing the attack surface and containing breaches [3]. Ultimately, ZTA provides a comprehensive security framework suitable for the decentralized, dynamic environment of edge computing, addressing specific vulnerabilities and enhancing the network's resilience against cyber threats.

Microservice architectures [4] are widely used in IoT systems that are inherently highly distributed and occasionally experience disconnections. By breaking down applications into independently deployable services, microservices can operate locally, close to the point of need. Microservices are independently deployable units, typically encompassing related functions. On the other hand, nanoservices are lightweight, single-purpose components that are optimized for dynamic deployment on resource-constrained IoT edge nodes [1].

Motivation. Modern edge computing systems are subject to an expanding array of demands. These systems must support low-latency execution, minimize energy usage, and enforce robust security. These requirements must be met within highly constrained IoT environments. Existing approaches frequently depend on centralized architectures, substantial containerization, or neural network-based inference pipelines. In some cases, these approaches assume the availability of powerful hardware, and in other cases, they focus on specific classes of workloads. These models are not easily adaptable to real-world deployments in which devices operate intermittently, possess limited memory and processing power, and must adapt dynamically to changing service demands. In this context, nanoservices are a promising alternative. The lightweight,

Simone Soderi is with IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy. He is also with the University of Padova and the Centre for Wireless Communications, University of Oulu, 90570 Oulu. email: simone.soderi@imtlucca.it.
Cinthya Celina Tamayo Gonzalez is with IMT School for Advanced Studies Lucca, Piazza San Francesco 19, 55100 Lucca, Italy. email: cinthya.tgonzalez@imtlucca.it.
Ijaz Ahmad and Erkki Harjula are with the Centre for Wireless Communications, University of Oulu, 90570 Oulu, Finland email: [ahmad.ijaz, erkki.harjula]@oulu.fi. This research is partially supported by the Finnish Doctoral Program Network in Artificial Intelligence, AI-DOC (decision number VN/3137/2024-OKM-6).

single-purpose structure of these systems facilitates dynamic deployment on devices with limited resources while concurrently enabling precise control over functionality and security. Specifically, nanoservices implementation enables the targeted integration of authentication, confidentiality, and integrity mechanisms directly at the physical layer. This integration serves to enhance the resilience to evolving threats [5].

This study proposes a nanoservice-level abstraction, designed to address these challenges by providing a lightweight and secure solution. This abstraction enables controlled deployment, memory-efficient execution, and dynamic security enforcement. Unlike conventional models that function at the task or workload level, our model is designed to facilitate generalized nanoservice orchestration while preserving adaptability, scalability, and low overhead. These characteristics position it as a practical foundation for secure and sustainable edge computing in future IoT and 6G systems.

Contribution. The key contributions of this paper are summarized as follows:

- We propose a nanoservice-based framework designed to support secure, lightweight, energy-efficient, and scalable operations in edge computing systems under constrained IoT environments.
- We integrate ZTA at the nanoservice level, enabling dynamic deployment and continuous enforcement of security policies directly on endpoints and edge nodes.
- We model the proposed framework using queueing theory. We represent endpoint and edge-level behaviour with $M/M/1$ and $M/M/m$ systems, respectively, and use this formulation to characterize nanoservice management and performance dynamics.
- We validate the proposed model through implementation on real IoT hardware, using MicroPython on ESP32 and Python on Raspberry Pi (RPi), demonstrating its feasibility on highly constrained edge devices.
- We demonstrated through a robust integration of mathematical analysis and experimental validation that the proposed model exhibits superior latency and energy efficiency performance compared with a centralized baseline.

Organization. This paper is organized as follows: Section II reviews prior work on edge computing and nanoservices. Section III introduces essential concepts, including edge/fog architectures and security/energy considerations in IoT. It also presents the proposed system architecture and the integration of nanoservices to enhance security and energy sustainability. Section IV analyzes the system's security, energy efficiency, and latency. Section V outlines the experimental setup. Section VI presents and discusses the results. Finally, Section VII summarizes the model and suggests future research directions.

II. Related Work

This section reviews prior work on lightweight virtualization and memory management in IoT networks. It also includes recent advances in energy-efficient edge paradigms to contextualize the contributions of our model within the broader landscape of dynamic and resource-aware edge solutions.

Harjula *et al.* [1] proposed a decentralized IoT edge nanoservice architecture aimed at future gadget-free computing. This architecture supports virtualized, on-demand service composition based on the hardware and software resources available at the user's current location. Despite the technology-agnostic concept, the study evaluated the nanoservice architecture with the container technology. Containers are, however, quite demanding in terms of computational resources, leaving room for optimizations in resource efficiency.

The concept of Femto-Containers [6], represents a novel approach to lightweight virtualization and fault isolation for small software functions on low-power IoT microcontrollers. Femto-Containers are designed to bridge the gap between the rudimentary Application Programming Interfaces (APIs) in IoT microcontrollers and the flexibility of networked software in less constrained environments. By enabling the secure deployment, execution, and isolation of small virtual software functions on low-power IoT devices over the network, Femto-Containers offer an attractive trade-off regarding memory footprint overhead, energy consumption, and security. Another remarkable work is Compressed Random Access Memory for Embedded Systems (CRAMEs), which presents an efficient software-based Random Access Memory (RAM) compression technique for embedded systems [7]. CRAMEs adjusts the size of the compressed RAM area, protecting applications capable of running without it from performance penalties.

In [8], Irtija et al. present a framework that integrates satisfaction games with approximate computing to optimize task offloading for IoT devices executing Deep Neural Network (DNN) workloads. Their approach utilizes Fully Autonomous Aerial Systems (FAAS) with heterogeneous DNN accelerators, enabling dynamic adaptation of computational precision based on user-defined Quality of Service (QoS) thresholds. The formulation of offloading as a satisfaction game demonstrated to enhance energy efficiency. However, [8] focuses primarily on deep learning workloads and assumes relatively powerful edge or aerial hardware is available.

Recent approaches have explored energy-efficient strategies for edge computing systems with advanced techniques. The work on [9] proposes an approximate computing framework that jointly optimizes compute, memory, sensor, and communication subsystems to reduce energy usage in edge inference pipelines. This framework facilitates dynamic quality adaptation across a range of Artificial Intelligence (AI) workloads. Another relevant work is presented in [10], which presents a predictive scheduling algorithm for heterogeneous multicore processors. This algorithm integrates deadline-aware task prioritization and core suitability estimation. The model accounts for distinct core types' performance and power characteristics.

While recent approaches based on satisfaction games and approximate computing have introduced promising mechanisms for energy efficiency, they primarily target specific workloads and rely on architectures with heterogeneous core configurations. Moreover, approximate computing may trade off fidelity to reduce energy consumption [11], whereas our model maintains functional accuracy by dynamically managing nanoservices as lightweight, self-contained units. This modular design enables selective deployment and local execution, allowing the system to adapt to constrained environments without compromising correctness. By supporting event-driven

service invocation and minimal memory overhead, the proposed framework offers a flexible and sustainable alternative for general-purpose edge computing scenarios.

Some studies have examined ZTA models for 6G networks. For instance, [2] proposed a software-defined ZTA for 6G, enabling secure access control and mitigating attacks. This proposal was validated through simulations of worm propagation and zero-day Distributed Denial of Service (DDoS) attacks. In [12], a ZTA framework for Unmanned Aerial Vehicles (UAVs) based 6G networks was introduced, leveraging federated learning and blockchain for UAVs operations and their digital twins. A real-time monitoring approach was shown in [13], featuring a learning-based ZTA that combines digital forensics and machine learning for adaptive security. [14] proposed a lightweight authentication protocol for 6G maritime systems, addressing maritime vulnerabilities.

In summary, previous studies have made substantial progress in the areas of lightweight virtualization, memory optimization, and energy-aware scheduling in edge computing. However, some of these methodologies demonstrate deficiencies in terms of modularity, execution granularity, and adaptability under constrained conditions. For instance, container-based nanoservice deployments [1] remain too resource-intensive for constrained nodes, while Femto-Containers [6] and memory compression techniques [7] address resource constraints but lack dynamic service orchestration. Energy-aware methods based on satisfaction games or approximate computing [8]–[10] often target specific workloads and may compromise computational fidelity. Moreover, these systems usually depend on neural network-based components, which require specialized hardware and are less well-suited to lightweight, general-purpose applications with resource-constrained nodes. In comparison, our model proposes a lightweight nanoservice-based system designed for constrained environments. It maintains functional correctness while enabling selective deployment and event-driven execution. Our proposed model enables adaptation to dynamic workloads and energy constraints, bridging key gaps in flexibility, security integration, and general-purpose applicability across diverse edge scenarios.

III. System Model

This section presents the security and energy considerations relevant to sixth-generation mobile networks. We introduce key concepts and explore the integration of nanoservices as a solution to enhance IoT system performance, security, and energy sustainability.

A. Edge and Fog network architecture

In the evolving landscape of networks towards 6G, our model adopts a comprehensive multi-layered approach for enhancing performance, security, and sustainability in IoT edge computing through nanoservices. As shown in Figure 1, this approach integrates cloud, edge and fog computing with IoT sensors, benefiting from the distributed nature of IoT devices and the computational strength available at the network's edge to improve security, minimize latency and boost efficiency overall. The Cloud tier offers vast computational resources and storage capacities, ideal for handling complex processing tasks and long-term storage [15]. However, the centralization inherent to cloud computing may result in latency and bandwidth bottlenecks, particularly impacting real-time operational needs. The Edge tier brings computational resources to the proximity of IoT devices to address latency challenges and resource efficiency. By performing data processing and decision-making tasks locally, edge servers alleviate the need for constant data transmission to the Cloud [16].

The Local tier establishes a decentralized virtualized local architecture to complement the edge tier. This tier may include highly resource-constrained nodes, such as sensor nodes, typically capable of hosting only one nanoservice, and less resource-constrained nodes like end-devices and gateways, called local edge nodes, capable of serving the local computation needs. Central to this paper is employing nanoservices,i.e., small, lightweight, and independent functionalities that can be dynamically deployed on resource-constrained devices and integrate with the rest of the IoT architecture (see Figure 1).

Long-Term Sustainability and Robustness. Our model has been designed to maintain both energy efficiency and security integrity over extended periods of operation, even under conditions of elevated or sustained workload. Adhering to an architectural paradigm characterized by the instantiation of nanoservices exclusively upon request and their subsequent elimination in periods of idleness, the proposed approach effectively mitigates service bloating and the long-term accumulation of inactive elements. Such accumulation is a known cause of memory saturation [17] and performance degradation in constrained environments [18]. In the context of ensuring the maintenance of a system's responsiveness, the employment of memory-aware execution, in conjunction with the periodic reassessment of service demand, is of fundamental importance. This approach also contributes to the reduction of energy cycles that are deemed superfluous over time.

Real-world Applicability. The proposed nanoservice-based approach (see Figure 1) can be applied to a diverse set of latency-sensitive and resource-constrained IoT domains. For instance, in smart healthcare, wearable devices and bedside monitors in hospitals can dynamically request authentication or data validation nanoservices from nearby edge nodes, enabling real-time patient monitoring with minimal delay and energy usage. In industrial automation, factory-floor sensors can invoke lightweight anomaly detection or device attestation nanoservices to detect faults and prevent production disruptions. Likewise, smart agriculture scenarios can leverage nanoservices for localized environmental data analysis or actuator control, ensuring timely interventions based on real-time sensing. These use cases benefit from our proposed model's event-driven nanoservice orchestration and energy-aware resource management, validating its versatility and effectiveness across practical edge computing scenarios.

Model Scability and Adaptability. Our system is a fully decentralized, event-driven deployment model that minimizes resource contention even as the number of connected devices grows into the hundreds or thousands. Each endpoint functions independently under an event-driven execution model, wherein nanoservices are triggered on demand and either executed

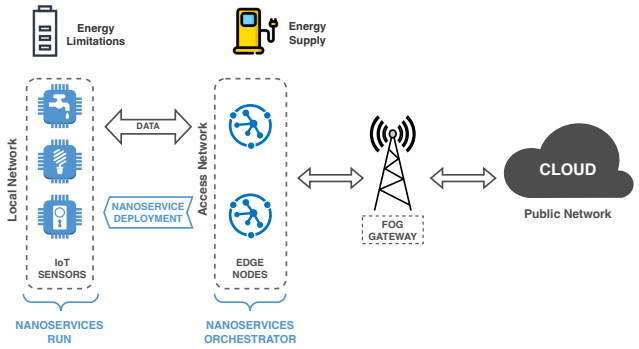


Fig. 1. Application scenario of nanoservices in an IoT edge computing network architecture.

from Non-Volatile Memory (NVM) or dynamically fetched from a proximate node, reducing redundant communication and memory overhead across the network. To address the issue of constrained local storage, the system implements a prioritization strategy for the retention of frequently accessed nanoservices. This improves long-term memory efficiency while also allowing for quick adaptation to changing nanoservice demand. Furthermore, the distribution of service orchestration across edge nodes, in conjunction with the facilitation of dynamic nanoservice exchange between endpoints and edge infrastructure, serves to mitigate central bottlenecks. This coordination is supported by queuing models that capture changing levels of request dynamics. In this manner, our model ensures balanced load distribution and sustained performance, even under growing network demand. The system minimizes the risk of service disruption during sudden demand surges through a combination of asynchronous request handling and fallback to alternative nodes when necessary.

B. Security and Energy sustainability of next-generation networks

One of the central goals for 6G networks is sustainability [19], which encompasses social, environmental, and economic aspects. The traffic amount will increase in future IoT deployments, increasing network congestion and demanding more computational resources, contributing to increased energy consumption. Energy-efficient computing paradigms are crucial for optimizing the processing, managing, and storing of this growing volume of data. In this context, sustainable security is essential for protecting the network and its users while ensuring its longevity and adaptability [20]. Security protocols must adapt as 6G develops to meet new technical demands, focusing on long-term resilience and resource efficiency.

Deploying security in ZTA by running nanoservices on resource-constrained IoT sensors, characterized by their reliance on microcontrollers with memory and computational capabilities, presents a formidable challenge. However, the integration of nanoservices, when paired with a ZTA, represents a viable strategy to mitigate these challenges. In the context of secure IoT sensors, nanoservices are designed to be lightweight and modular. This design philosophy significantly reduces their footprint. Moreover, applying zero-trust principles directly within each nanoservice ensures they can be individually secured, thereby reducing the potential impact on the entire system should any single service be compromised. ZTA applied to nanoservices involves critical measures: mutual authentication establishes trust between sensors and the edge nodes using protocols like pre-shared keys. At the same time, the principle of least privilege limits each nanoservice's permissions, reducing unauthorized access risks. Communication security is ensured through encryption, protecting data from eavesdropping. Nanoservices stored in the sensor's NVM are secured with encryption and access controls, and secure boot procedures verify that only authorized code runs on startup, preventing malicious activities.

Algorithm 1: Nanoservice Event-Driven Request Handling in an IoT Sensor

procedure NANOSERVICE REQUEST
  Input: Request for a nanoservice N
  Check if running:
    If nanoservice N is running, the process ends immediately.
  If not running:
    List the nanoservices currently in NVM.
    Check if N is stored in NVM.
  NVM Check:
    If stored: Load N from NVM and run it.
    Process ends.
    If not stored: Request N from the edge node.
  NVM Capacity Check: ▷ Check if NVM is full.
    If full: Replace the least recently used nanoservice.
    Copy N from the edge node to NVM.
  Run Nanoservice: ▷ Load N from NVM
    Run the nanoservice N.
    Process ends.
  Output: Nanoservice N is now executed.
end procedure

We assumed IoT sensors to implement an event-driven model for reacting to incoming requests for specific nanoservices, as depicted in Algorithm 1. Upon receiving a request, the sensor first checks if the nanoservice is running locally or stored in the local NVM. If found from either, it is directly executed. Otherwise, the sensor initiates remote execution by requesting the edge node. The edge node performs security checks, retrieves the nanoservice, and deploys it to the sensor's NVM if there is enough memory to store it. After deployment or local availability, the service is loaded and executed.

IV. Studied Problem and Proposed Framework

The problem that is addressed in this work is the secure and efficient execution of nanoservices in resource-constrained IoT edge environments. The objective of this study is to minimize latency and energy consumption associated with service deployment and management while offering a secure solution.. Achieving this optimization necessitates the consideration of several constraints, including the limited availability of memory and computing resources at endpoints, the dynamic character of service request patterns, and the continuous need for security enforcement. Furthermore, it is crucial that the system can support adaptive nanoservice orchestration without reliance on centralized infrastructure. The edge computing architecture comprises many components subjected to many request types at varying frequencies. Applying queue theory

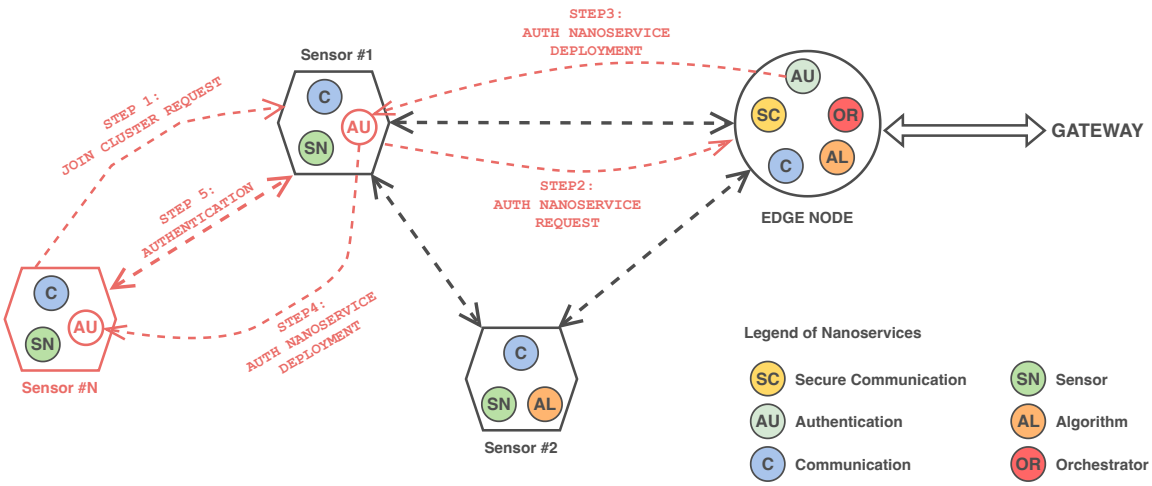


Fig. 2. Nanoservice mechanism deployment when a new sensor joins the local cluster. The decentralized identity authentication use case.

[21] allows for systematically modelling this architecture's capabilities and request patterns. We assume service arrivals follow a Poisson distribution and service times are exponentially distributed. Each service at the end-points can be modelled as a single $M/M/1$ queue, where λ_i denotes the arrival rate and μ_i represents the service rate for service i. In contrast, the edge node does not face strict hardware limitations and can handle multiple requests simultaneously. Consequently, the edge node can be represented as an $M/M/m$ queue, where m represents the number of different services it can handle concurrently.

To evaluate the proposed model, we consider a scenario involving several nanoservices. The model incorporates memory and energy constraints, ensuring that each endpoint retains only the most frequently requested services. If a node requires a nanoservice not currently in its memory, it can request it from another node. Let us denote the most frequently requested services as S_1, S_2, ..., S_Q. On the other hand, the services which are less frequently requested are denoted as S_{Q+1}, S_{Q+2}, ..., S_K. As a result, the arrival rates associated with the frequently accessed nanoservices are markedly higher than those associated with the less frequently requested nanoservices ($\lambda_{S1}, \lambda_{S2}, \ldots, \lambda_{SQ} \gg \lambda_{SQ+1}, \lambda_{SQ+2} \ldots \lambda_K$).

In the following analysis, the performance of our model will be contextualized by comparing it to a classical centralized edge computing framework [22]. In this centralized framework, services are statically hosted in a central cloud or upper-tier node. In such cases, all service requests from endpoints are routed through multiple hops to the cloud, leading to increased latency and energy consumption, particularly under conditions of frequent or repetitive service demands [23]. This approach is short on dynamic service allocation and local adaptability, which makes it a relevant baseline for evaluating the benefits of decentralized, nanoservice-based approach.

A. Latency Analysis

Let us define g as the number of hops required from the requesting endpoint to the edge node and comeback. Similarly, let us define r as the number of hops from the endpoint to the cloud and comeback to the endpoint, including also the fog gateway. Given that q encompasses the hops between the edge node and the cloud, we can conclude that $g < r$.

Two parameters must be considered: the hop time (t_{hop}), which denotes the time required for a single hop, and the waiting time (W) at the node or server delivering the nanoservice. It is important to note that the waiting time encompasses both the queueing delay and the service time. To estimate this, the queueing formula $L = \lambda W$ [24] can be applied, where L is the average number of customers in the node or server. The value of L is contingent upon the behaviour of the network and the selected queueing model. For simplicity, we will assume that the system operates under equilibrium conditions. In the case of an $M/M/1$ queueing model, the average number of customers (L) can be computed as $L = \frac{\lambda}{\mu - \lambda}$ [25]. Where μ denotes the mean service rate. Consequently, the average waiting time can be calculated as $W = \frac{1}{\mu - \lambda}$. On the other hand, For a $M/M/m$ model, where m represents the number of parallel servers, the average number of customers is derived as $L = L_o + \frac{\lambda}{\mu}$ [26], where L_o represents the average number of customers waiting in the queue. Therefore, the average waiting time is computed as $W = \frac{L_o}{\lambda} + \frac{\lambda}{\mu}$.

Considering the previous definitions, the latency for the proposed model's local services L_{loc} is calculated as follows:

$$L_{\text{Loc}} = 0 \cdot t_{\text{hop}} + W = \frac{1}{\mu - \sum_{i=1}^{Q} \lambda_{S_i}} \tag{1}$$

The latency when the nanoservice is requested to the edge node L_{Edge} is as follows:

$$L_{\text{Edge}} = g \cdot t_{\text{hop}} + W = g \cdot t_{\text{hop}} + \frac{L_o}{\sum_{i=Q+1}^{K} \lambda_{S_i}} + \frac{\sum_{i=Q+1}^{K} \lambda_{S_i}}{\mu} \tag{2}$$

Regarding the centralized model, the latency L_{Cen} can be expressed as follows:

$$L_{\text{Cen}} = r \cdot t_{\text{hop}} + W = r \cdot t_{\text{hop}} + \frac{L_o}{\lambda_T} + \frac{\lambda_T}{\mu} \tag{3}$$

Where $\lambda_T = \sum_{i=1}^{K} \lambda_{S_i}$. To compare the performance, the weighted latency is calculated based on the arrival rates.

Considering 1 and 2, the weighted latency for the proposed model L_{PW} is calculated as follows:

$$L_{\text{PW}} = \left(\frac{\sum_{i=1}^{Q} \lambda_{S_i}}{\lambda_T} \right) L_{\text{Loc}} + \left(\frac{\sum_{i=Q+1}^{K} \lambda_{S_i}}{\lambda_T} \right) L_{\text{Edge}} \quad (4)$$

The overall weighted latency for the centralized model L_{CW} can be mathematically expressed as follows:

$$L_{\text{CW}} = \left(\frac{\sum_{i=1}^{K} \lambda_{S_i}}{\lambda_T} \right) \left(r \cdot t_{\text{hop}} + \frac{L_o}{\lambda_T} + \frac{\lambda_T}{\mu} \right) \quad (5)$$

$$\stackrel{(a)}{=} r \cdot t_{\text{hop}} + \frac{L_o}{\lambda_T} + \frac{\lambda_T}{\mu} \quad (6)$$

where (a) follows the simplification of the weighting factor according to the definition of λ_T. To demonstrate the efficiency of the proposed model, the difference between latencies (ΔL) must be positive:

$$\Delta L = L_{\text{CW}} - L_{\text{PW}}. \quad (7)$$

A more thorough examination is necessary to ascertain the behaviour of 7. In the centralized model, all services experience uniform latency, which it consistently higher due to $g < r$. In the proposed model, the latency due to the locally stored services 1 is considerable lower than the latency required for this services in the centralized model. Additionally, since $\frac{\sum_{i=1}^{Q} \lambda_{S_i}}{\lambda_T} \gg \frac{\sum_{i=Q+1}^{K} \lambda_{S_i}}{\lambda_T}$, the impact of the higher latency term for edge node-assisted services in L_{PW} is limited compared to the centralized model's uniform higher latency. Therefore, we can conclude $\Delta L \gg 0$.

B. Energy Consumption Analysis

A multi-tier network comprises devices with different resource consumption across varying tiers. For instance, it has been shown that devices in the cloud tier exhibit higher energy consumption compared to those in the endpoint tier [27]. To generalize the energy consumption across all devices used within the tiers, we define the variable P_A as the average power consumed by the network. Given the equation 4, the energy consumed for the proposed model can be defined as:

$$E_{\text{P}} = \left(\frac{\sum_{i=1}^{Q} \lambda_{S_i}}{\lambda_T} \cdot L_{\text{Loc}} + \frac{\sum_{i=Q+1}^{K} \lambda_{S_i}}{\lambda_T} \cdot L_{\text{Edge}} \right) P_A \quad (8)$$

Similarly, according to the equation 7, the energy consumed by the classical centralized model can be expressed as follows:

$$E_C = \left(r \cdot t_{\text{hop}} + \frac{L_o}{\lambda_T} + \frac{\lambda_T}{\mu} \right) P_A \quad (9)$$

To demonstrate energy efficiency, the differences between equations 8 and 9 expressed as $\Delta E = E_P - E_C$ must be positive. Analyzing the equation by its elements, we first consider the time needed to request a nanoservice. In our model, the requesting device is closer to the service provider in any type of service, as $g < r$. Furthermore, it has been demonstrated that the total weighted latency, which accounts for request frequency and the serving queue, is lower in the proposed model compared to the centralized model. The impact of power in the equation can be found to be negligible in this disparity. Consequently, it is concluded that $\Delta E \gg 0$ and there is an energy efficiency.

To provide a representation of the latency and energy performance of the two models (equations 9, 8, 6, 4), their behaviour is illustrated in Figure 3. Let us exemplify with a system system inspired by the requirements outlined in [28]. The network uses three nanoservices with more frequency ($\lambda_{S1,S2,S3} = 57,600$ requests per day) and two nanoservices with less frequency ($\lambda_{S4,S5} = 5,760$ requests per day), using $r = 4$ and $g = 2$. The average energy consumed in each tier is 6.3[J], from the values presented in [27] and $t_{hop} = 0.01$ s. To better understand the impact of factor L_o, the performance is presented with two different values, one being 25% higher than the other. As shown in Figure 3, both models show reduced response times as service rates increase. In contrast, as the demand for requests in the queue rises, as shown by L_o, the latency in both models rises as well. However, our model substantially outperforms the centralized model, demonstrating greater efficiency. Analogously, it can be seen that the energy consumed is considerably lower in our model.

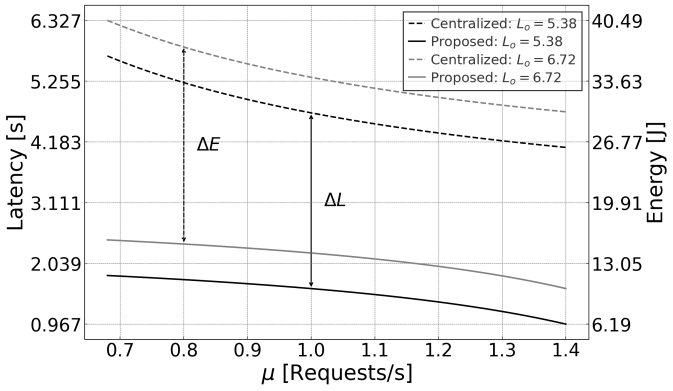


Fig. 3. Latency (L) and energy (E) comparison for centralized and proposed models for different average number of requests in the system (L_o).

C. Resilience and Zero Trust as the Security Framework

In edge computing, data generation and processing occur close to its source [1]. Proximity also reduces latency by minimizing the distance between the points of computation and the endpoints in question whilst simultaneously reducing the overall volume of sensitive data transmitted across the network. This, in turn, reduces the overall exposure to potential risks. Edge computing systems may exhibit dynamic and time-dependent behaviours [29], necessitating the adaptation of security frameworks to accommodate evolving requirements and interactions. The proposed model addresses this challenge by enabling the transfer of nanoservices as required based on the needs of the endpoints. Such nanoservices, including those about security, can be conveyed and exchanged dynamically between endpoints and edge nodes. This feature guarantees the provision of security solutions capable of adapting to the evolving demands and the emergence of new threats.

Moreover, despite the common perception of the cloud as an environment with ample resources to support sophisticated protocols, it is crucial to acknowledge the energy and capacity constraints of endpoints. Consequently, security strategies at the endpoint level must be lightweight.

To facilitate scalability in ZTA implementations, our model adopts a decentralized nanoservice-based design. Authentication and access control functionalities are delegated to lightweight nanoservices operating across edge nodes and capable endpoints, minimizing reliance on centralized authorities. Additionally, verification is managed in an event-driven mode, triggered by specific service requests rather than periodic polling. This approach reduces communication and processing overhead. Accordingly, the model facilitates uninterrupted operation under conditions of elevated load, such as bottlenecks, thereby enhancing system responsiveness and stability.

One notable characteristic of this category of networks is the potential for endpoints to intermittently disconnect and reconnect to the network, contingent on the specific application in use. Our model accommodates transient behaviour while maintaining compliance with a ZTA. Following this approach, each element must undergo individual authentication, including endpoints. ZTA facilitates the detection of anomalous nodes by treating entities performing nanoservices as untrusted by default. Our model presents a framework that can withstand potential attacks, adapt to evolving network conditions, and enforce rigorous security checks at each interaction point.

Concerning security-related matters, the existing literature offers several pertinent examples. For example, [30] proposes a collaborative and hierarchical detection system for edge computing systems based on ZTA. It emphasizes collaboration between network elements, with signature-based detection as the sole dynamic response method. However, this model is constrained by its dependence on hierarchical structures, in which only devices located at higher levels of the hierarchy can assist. In contrast, our proposed system employs a more flexible security framework, enabling any device to contribute. In opposition to this approach, the system LocKedge proposed in [31] presents a multi-attack detection model implemented at the edge layer. This model is designed to capitalize on the capabilities of edge nodes while facilitating a prompt response to potential attacks. [31] shows low complexity of data analysis and rapid detection of attacks. Nevertheless, the success and performance of the model are largely contingent upon the physical attributes and constraints of the edge node.

Femto-Containers [6] represent a lightweight virtualization solution tailored for the IoT microcontrollers. This approach offers several features, including secure isolation and memory access control. However, the system lacks adaptable security mechanisms capable of dynamically responding to evolving network needs. Moreover, [6] lacks a cooperative security framework in which devices would collaborate to enhance the system's overall security. In [2], a collaborative security system for 6G networks was proposed. The system employs a software-defined ZTA to mitigate potential threats. The system incorporates security protocols provided by blockchain-based third-party services. However, this solution was not designed considering resource-constrained devices.

TABLE I
COMPARISON OF SECURITY MODELS.

	Sedjelmaci et al. [30]	LocKedge [31]	Femto Containers [6]	Chen et al. [2]	Proposed Model
Lightweight Security Service	✓	✓	✓	✓	✓
Collaborative Security Frame	✓	✓		✓	✓
Resilient Against Many Attacks		✓	✓	✓	✓
Adaptability to Network Needs	✓	✓		✓	✓
ZTA Over Edge Computing	✓			✓	✓
Resources Limited Devices			✓		✓
Latency Analysis	✓		✓		✓
Energy Analysis					✓
Nanoservice Deployment					✓

Table I compares the essential security components addressed in the existing literature and our model. This comparison illustrates the comprehensive security capabilities of the proposed model. In contrast to the existing literature, our model incorporates all the security aspects covered by previous studies and ensures secure nanoservice deployment. Furthermore, our model is the only work that incorporates an energy efficiency analysis tailored to a ZTA in devices with limited resources.

V. EXPERIMENTAL SETUP

This section presents the findings of our experiments, which were conducted using the Python and MicroPython environments with all services implemented in the bytecode format. In the initial experiment, we constructed a scenario where an endpoint requires a service and requests it from a nearby edge node. This experiment utilised a RPi model 4 as the edge node, while an ESP32 WROOM-32 was the endpoint. We developed three nanoservices, designated A, B, and C, with varying sizes. The size of Nanoservice A is 2.81 kB, that of Nanoservice B is 28.48 kB, and that of Nanoservice C is 55.87 kB.

To commence the experiment, the RPi and ESP32 were connected to the internet and synchronised their clocks. Following the synchronisation process, a Bluetooth Low Energy (BLE) connection was established between the devices, whereby the ESP32 transmitted a request to the RPi. An acknowledgement mechanism was introduced to guarantee the reliability of data transmission. The endpoint duly acknowledged each packet received, thus enabling the edge node to retransmit any packets lost or unacknowledged packets. Furthermore, the ESP32 transmitted status updates at two-second intervals to the RPi until it received an End of File (EOF) indicator, signifying the successful transfer of the nanoservice. During the transmission, the ESP32 periodically communicates if the EOF flag has not yet been received and that the file transfer should continue. To better illustrate the viability and efficacy of our proposed model, we have also quantified the pertinent metrics associated with the fundamental activities that are

TABLE II
TRANSMISSION PROCESS AND TIMING SUMMARY.

Step	Action	Data Size [B]	Rate [kbps]	Time per Action [ms]	Repetitions	Subtotal [ms]
1	RPi transmits packet	32	688.47	0.372	140	52.08
	ESP32 receives packet and waits	-	-	0.150	140	21.00
	ESP32 sends acknowledgment	26	46.9	4.435	140	620.9
	RPi waits	-	-	100.150	140	14021.00
	Subtotal for Step 1 (ST_1)					**14714.98**
2	RPi transmits partial payload	25	688.47	0.290	1	0.29
	ESP32 receives packet and waits	-	-	0.150	1	0.15
	ESP32 sends acknowledgment	26	46.9	4.435	1	4.44
	RPi waits	-	-	100.150	1	100.15
	Subtotal for Step 2 (ST_2)					**105.03**
3	RPi transmits EOF flag	15	688.47	0.174	1	0.17
	ESP32 receives EOF flag and waits	-	-	0.150	1	0.15
	ESP32 sends acknowledgment	26	46.9	4.435	1	4.44
	Subtotal for Step 3 (ST_3)					**4.76**
E	ESP32 transmits stabilization messages	28	46.9	4.776	7	33.43
	RPi waits	-	-	0.150	7	1.05
	Subtotal for Periodic Interruptions (ST_E)					**34.48**
	Total Transmission Time for File A					**14859.25**

integral to our system. In the model, a requested service is transmitted to a node, stored in memory, loaded for execution, and eventually deleted from memory when no longer required.

A. End-to-End Latency Analysis

To establish a benchmark for the experimental results of our system, we have computed the theoretical transmission time based on the characteristics of BLE and our system implementation. The BLE protocol is designed to provide a theoretical data transfer rate of 1 Mbps [32]. However, the actual throughput observed in IoT devices can vary considerably due to various factors [33], resulting in notable discrepancies between devices. The time required by each device to transmit a single packet was measured to provide a more accurate estimation of the transmission performance. Subsequently, the aforementioned time was divided by the number of data transmitted to approximate the actual transmission capacity. Each packet was measured while transmitting the largest file, Nanoservice C. We observed that the RPi exhibits an actual throughput of approximately 688.47 kbps, whereas ESP32 achieves a throughput of 46.9 kbps. In consideration of the aforementioned findings, we proceed with the time analysis. In the first version of BLE, the payload size for each packet was limited to a maximum of 23 B [34], and to ensure compatibility with all different versions, the payload size was maintained within this limit: in 20 B per packet.

To show the calculations, we examine a scenario in which File A is transferred from a RPi to an ESP32 via BLE. The process yields 140 packets of full payload and a packet with partial payload. In addition to the payload, each packet contains 13 B of overhead. The overhead amount was corroborated through analysis with the program Wireshark. Table II provides a comprehensive account of the data transmission, delineating the flow of communication and the time associated with each step. The transmission is divided into three principal phases: full payload transmission, partial payload transmission, and the EOF flag transmission. The transmission time of the periodic messages is also considered.

In the initial stage, the RPi transmits a series of packets containing a data payload of 20 B to the ESP32. It receives the data for each packet and then pauses briefly before sending an acknowledgement. Additionally, the RPi incorporates a waiting period between successive transmissions. This phase is repeated for 140. The second phase entails the transmission of a partial payload, whereby the RPi transmits a single packet with 13 B of payload. As with the preceding phase, the ESP32 acknowledges the packet and includes a waiting time. Finally, an EOF flag is transmitted. Concurrent with these phases, the ESP32 periodically transmits messages containing status updates. These messages are transmitted at fixed intervals, each message followed by a waiting period on the RPi. Ultimately, the summation of all timing subtotals for each phase (ST_1, ST_2, ST_3) and for periodic interruptions (ST_E) yields the total transmission time of 14.86 s.

The experimental time was measured using both devices' Real Time Clock (RTC). Before transmitting the initial packet, the edge node recorded the RTC timestamp. Upon receipt of the EOF flag, the ESP32 recorded its RTC timestamp. The total transmission time was calculated as the difference between the aforementioned timestamps. To ensure the consistency and reliability of the results, each nanoservice transfer experiment was repeated five times. In the experiment, the endpoint is the node requesting a nanoservice; thus, the time measurements for the key activities after file transmission were conducted on the ESP32: saving the code in memory, which includes the allocation of the received service into NVM; loading from memory, which involves retrieving the service from memory for execution without including the execution itself; and removing the nanoservice, which measures the time taken to erase a nanoservice from NVM once it is no longer needed.

B. Energy Consumption Testbed

This subsection extends our time analysis by rigorously examining energy consumption, a critical factor for battery-operated IoT devices. We employed the Otii Ace Pro device

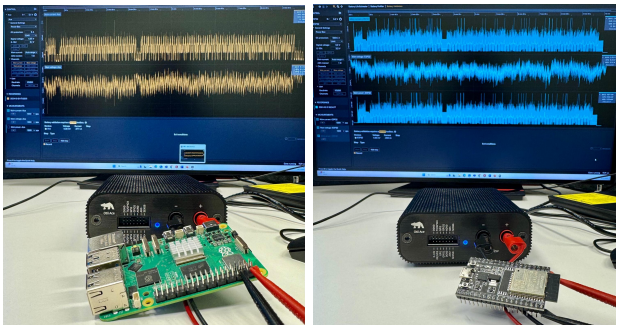


Fig. 4. RPi (Left) and ESP32 (Right) Energy Measurements with Otii.

(Fig. 4), which accurately records voltage (V), current (I), power (P), and energy (E). The ESP32 nodes and the RPi were powered through the Otii Ace Pro, allowing us to measure V and I. We then computed P and E using $E = \sum_{n=1}^{N} P_n \cdot \Delta t = \sum_{n=1}^{N} (V_n \cdot I_n) \cdot \Delta t$, where V is in volts, I in amperes, P in watts, and E in joules. For ESP32 devices, we configured Otii Ace Pro with 3.3 V and 500 mA overcurrent protection. We used 5.1 V and 5 A settings for the RPi. All experiments were repeated, and time and energy measurements were recorded for analysis. The results are presented in Section VI.

VI. Evaluation and Results

1) File Transfer Latency: Figure 5 presents the theoretical and experimental results for the time required to transmit Nanoservices A, B, and C from the edge node to the endpoint. The results show that transmission time increases in direct proportion to the file size, with larger files necessitating longer transmission times. It is also noteworthy that, as expected, the experimental transmission times exceed the theoretical predictions across all cases. The discrepancy observed between the theoretical and experimental results is approximately 20% for all three nanoservices, indicating that the performance of the experimental model aligns reasonably well with the theoretical estimations. This consistency is further evidenced by the standard deviation, which remains below 9% in all cases. In particular, the standard deviation for B and C is approximately 2%, whereas for A, it is around 8.5%.

2) Time measured during Save, Delete, Loading a Nanoservice: Table III presents the performance of the key operations

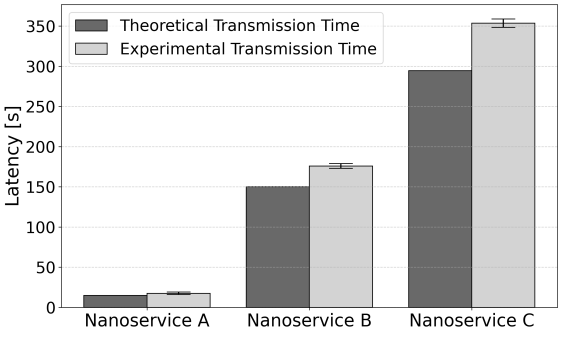


Fig. 5. Comparison of theoretical and experimental total transmission time for the three nanoservices.

TABLE III
ESP32 Time measurements for save, delete, and load cycles.

Nanoservice \ Operation	Save		Delete		Load	
	$\mu_T[ms]$	$\sigma_T[ms]$	$\mu_T[ms]$	$\sigma_T[ms]$	$\mu_T[ms]$	$\sigma_T[ms]$
Nanoservice A	95.08	189.47	11.79	138.4	2167.6	559.51
Nanoservice B	573.8	183.65	34.35	192.24	4682.5	535.47
Nanoservice C	1135	199.48	57.51	254.53	6510.1	523.89

in our model, elucidating the relationship between nanoservice size and the time invested for each operation with the three files. The average time (μ_T) and standard deviation (σ_T) required for the saving operation increased proportionally to the nanoservice size. Specifically, Nanoservice A exhibited the shortest save time (95.08 ms), followed by Nanoservice B (573.8 ms) and Nanoservice C (1.14 s). This trend suggests that the resource demands are directly proportional to the size of the nanoservice. The results indicate a variability in time measured, as evidenced by the standard deviation observed across different trials. Such variability can be attributed to many factors, including the reception and transmission capacities of the devices, which may be subject to alteration due to concurrent internal operations [35], [36].

These findings underscore the considerable influence of practical constraints and device limitations on transmission times, impacting overall system performance. Nevertheless, the results also validate the feasibility of our model and demonstrate its capacity to closely approximate theoretical predictions under experimental conditions. Even when the same device is employed, fluctuations in the behaviour of the internal clock of IoT devices can be observed [37], resulting in discrepancies in the time spent by the device waiting to receive messages and in the exact moment the clock is invoked.

The removal operation consistently reduced the required time compared to the saving and loading operations. The average removal time for Nanoservice A was 11.79 ms, while that for Nanoservice C was 57.5 ms. Compared to the saving operation, the standard deviation for removal exhibited a notable increase with file size, reaching 254.53 ms for the largest nanoservice. The loading operation exhibited the highest average times across all nanoservices, with Nanoservice C demonstrating the most extensive duration (6.51 s), followed by Nanoservice B (4.68 s) and Nanoservice A (2.16 s). These results reflect the considerable effort required for loading, particularly as nanoservice complexity increases.

As illustrated in Fig. 6, a comparative analysis of the total end-to-end latency between our model and a classical centralized framework is presented, considering nanoservices of varying size. For the smallest service (Nanoservice A), our model achieves a latency reduction of approximately 66.5 % compared to the centralized approach. For the largest service (Nanoservice C), the reduction reaches approximately 77.1 %. These results highlight the efficacy of our model in decreasing latency across the different nanoservices. Moreover, this latency reduction contributes to a more efficient use of energy.

Real-world Use Case: A practical real-world application of the proposed approach can be observed in hospital settings. For instance, remote patient monitoring systems in the Intensive

Care Unit (ICU) rely on continuous authentication between resource-constrained Internet of Medical Things (IoMT) devices. A nurse might undergo five authentication procedures to access patient data at the bedside. Moreover, these devices authenticate one another to ensure secure data exchange. By offloading the authentication nanoservice to a nearby node, the system significantly reduces latency and energy usage while maintaining zero trust compliance.

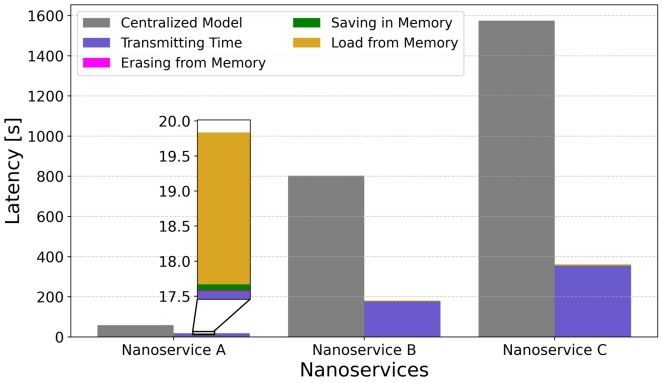


Fig. 6. Latency comparison between centralized model and proposed model across the management cycle.

3) Energy Consumption during File Transfer: Fig. 7 describes the power consumed by ESP32 device while importing the Nanoservices A, B, and C, respectively. During the experiment, as can be observed from the graph, each nanoservice is imported with an intermediate delay of 10 s to distinguish the results from each other. Similarly, Fig. 8 describes the power consumption pattern for the RPi device while exporting the nanoservices via BLE to the ESP32 device. The detailed results of this experiment are presented in Table IV along with Total End-to-End (E2E) Energy Consumption for each nanoservice along with idle and wait by all the participating devices, where μ_P represents the mean power value in watts, σ_P represents the standard deviation for the power values and E represents the energy consumed in joules. The results show a consistent increase from Nanoservice A to Nanoservice B and then ultimately to Nanoservice C. The primary reason for this increase is the increasing size of the respective nanoservice. It can also be seen from Fig. 7, Fig. 8, and Table IV that the energy consumption of RPi is the main contributing factor to these values compared to the energy consumed by ESP32 nodes. The primary reason for this is that the operating voltage of RPi is higher than that of ESP32.

TABLE IV
POWER AND ENERGY CONSUMPTION FOR DIFFERENT PHASES.

Process \ Phase	ESP32 (Receiver)			RPi (Sender)			Total E [J]
	μ_P[W]	σ_P[W]	E[J]	μ_P[W]	σ_P[W]	E[J]	
Nanoservice A	0.20	0.06	4.11	3.49	0.32	84.12	88.23
Nanoservice B	0.21	0.05	37.82	3.51	0.32	620.59	658.41
Nanoservice C	0.21	0.05	74.52	3.55	0.33	1241.93	1316.45
Idle	0.13	0.00	2.96	3.13	0.19	18.52	21.48
Waiting	0.16	0.04	5.58	3.09	0.23	135.51	141.08

4) Energy Consumption during Save, Delete, Loading a Nanoservice: Once the nanoservice is received by the ESP32, it can be stored in the NVM. We have computed the energy consumed by ESP32 while saving the nanoservice (A, B, C). The results are presented in Table. V. Similarly, the energy consumed in removing a particular nanoservice from the NVM of the ESP32 is also measured and presented in Table V. Finally, the energy consumed to load and/or unload a nanoservice in the ESP32 is also measured and presented in Table V. The measurements reveal that the cumulative energy required to save, delete, and load a nanoservice increases from Nanoservice A to B and C. These results are demonstrated in Fig. 9 and Fig. 10 respectively.

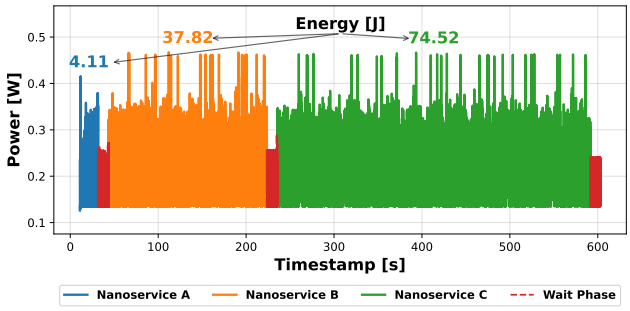


Fig. 7. Energy consumption graphs for ESP32 during file transfers.

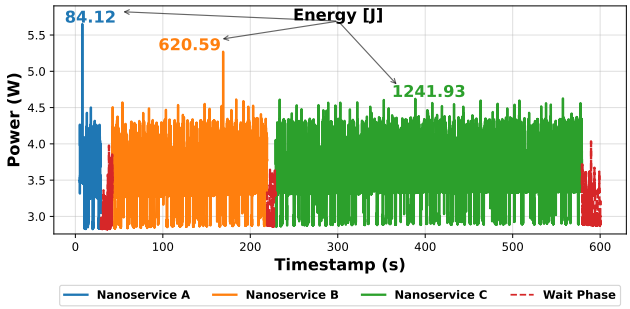


Fig. 8. Energy consumption graphs for RPi during file transfers.

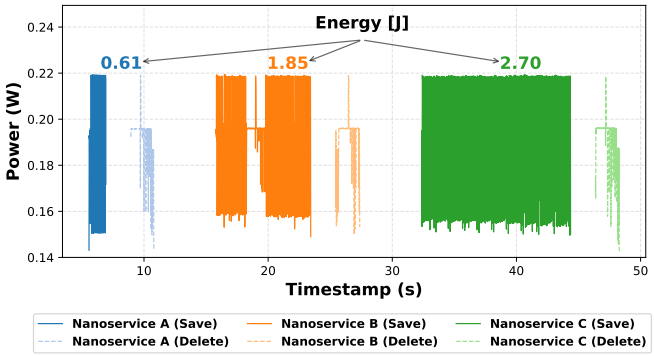


Fig. 9. Energy Consumption for ESP32 during nanoservice save/delete.

VII. CONCLUSIONS

This work presents a novel model designed to address the critical challenges of resilience, security, and sustainability in the evolution of 6G networks. The model is based on

TABLE V
POWER AND ENERGY CONSUMPTION OF ESP32 DURING COPY, DELETE, AND LOAD/UNLOAD OPERATIONS FOR DIFFERENT NANOSERVICES.

Operation / Nanoservice	Save			Delete			Load			Total E [J]
	$\mu_P[W]$	$\sigma_P[W]$	$E[J]$	$\mu_P[W]$	$\sigma_P[W]$	$E[J]$	$\mu_P[W]$	$\sigma_P[W]$	$E[J]$	
Nanoservice A	0.1886	0.0211	0.2584	0.1917	0.0112	0.3565	0.1310	0.0076	6.7677	7.3826
Nanoservice B	0.1935	0.0200	1.4780	0.1911	0.0122	0.3726	0.1317	0.0098	6.8519	8.7025
Nanoservice C	0.1944	0.0229	2.3295	0.1911	0.0116	0.3670	0.1344	0.0164	7.0510	9.7475

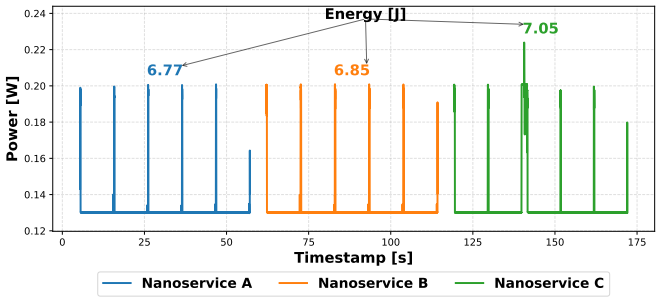


Fig. 10. ESP32 energy consumption during service load/unload.

a cooperative system wherein edge nodes communicate and collaborate to form an adaptive network. This adaptability enables the system to meet various network requirements while maintaining robust security by implementing a ZTA. Combining these features provides a prospective solution for the secure and sustainable operation of the IoT in 6G environments.

A comprehensive mathematical analysis was conducted to validate the model, focusing on key performance indicators such as latency and energy efficiency. This rigorous evaluation proved that the model offers a sustainable alternative to traditional approaches. The proposed model was implemented in its entirety using MicroPython and evaluated on real constrained devices (ESP32 and RPi), confirming its feasibility for deployment in practical IoT edge environments without the need for specialized hardware or inference accelerators. The experimental results demonstrated that the proposed model can reduce total service latency by up to 77 % in comparison with a centralized baseline. Consequently, the model exhibits enhanced responsiveness and superior energy efficiency, attributable to reduced active execution times.

While our model demonstrates optimistic results, it is important to note that it is subject to several limitations. The queueing models that have been adopted presuppose specific conditions (e.g., Poisson arrivals), which may not completely capture the variability of workload in reality. Moreover, while the framework exhibits energy efficiency, it uniformly processes all nanoservices without accounting for their criticality or contextual priority. Finally, our model operates under the assumption of a uniform execution environment across devices, such as consistent programming languages. This assumption may impose limitations on interoperability and increase memory requirements at the edge.

Future work will extend this research to broader test cases and alternative queuing models, supporting a more comprehensive investigation of IoT system behaviour under diverse conditions, moving beyond the $M/M/s$ and $M/M/1$ paradigms. Future efforts will also focus on assessing potential efficiency gains across different application contexts, alongside a comprehensive evaluation of advanced security mechanisms.

REFERENCES

[1] E. Harjula, P. Karhula, J. Islam, T. Leppänen, A. Manzoor, M. Liyanage, J. Chauhan, T. Kumar, I. Ahmad, and M. Ylianttila, "Decentralized iot edge nanoservice architecture for future gadget-free computing," *IEEE Access*, vol. 7, pp. 119 856–119 872, 2019.

[2] X. Chen, W. Feng, N. Ge, and Y. Zhang, "Zero trust architecture for 6g security," *IEEE Network*, vol. 38, no. 4, pp. 224–232, 2024.

[3] K. Ramezanpour and J. Jagannath, "Intelligent zero trust architecture for 5G/6G networks: Principles, challenges, and the role of machine learning in the context of O-RAN," *Computer Networks*, vol. 217, p. 109358, 2022.

[4] R. Laigner, Y. Zhou, and M. A. V. Salles, "A distributed database system for event-based microservices," in *Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems*, ser. DEBS '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 25–30. [Online]. Available: https://doi.org/10.1145/3465480.3466919

[5] D. D. Sánchez-Gallegos, A. Galaviz-Mosqueda, J. L. Gonzalez-Compean, S. Villarreal-Reyes, A. E. Perez-Ramos, D. Carrizales-Espinoza, and J. Carretero, "On the continuous processing of health data in edge-fog-cloud computing by using micro/nanoservice composition," *IEEE Access*, vol. 8, pp. 120 255–120 281, 2020.

[6] K. Zandberg, E. Baccelli, S. Yuan, F. Besson, and J.-P. Talpin, "Femto-containers: lightweight virtualization and fault isolation for small software functions on low-power iot microcontrollers," in *Proceedings of the 23rd ACM/IFIP International Middleware Conference*, ser. Middleware '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 161–173.

[7] L. Yang, R. P. Dick, H. Lekatsas, and S. Chakradhar, "CRAMES: compressed RAM for embedded systems," in *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 93–98.

[8] N. Irtija, I. Anagnostopoulos, G. Zervakis, E. E. Tsiropoulou, H. Amrouch, and J. Henkel, "Energy efficient edge computing enabled by satisfaction games and approximate computing," *IEEE Transactions on Green Communications and Networking*, vol. 6, no. 1, pp. 281–294, 2022.

[9] S. K. Ghosh, A. Raha, and V. Raghunathan, "Energy-efficient approximate edge inference systems," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 4, pp. 1–50, 2023.

[10] Y. Liu, H. Qu, S. Chen, and X. Feng, "Energy efficient task scheduling for heterogeneous multicore processors in edge computing," *Scientific Reports*, vol. 15, no. 1, p. 11819, 2025.

[11] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.

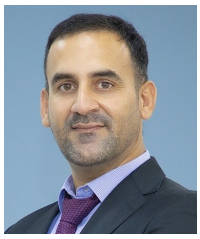
[12] I. A. Ridhawi and M. Aloqaily, "Zero-trust uav-enabled and dt-supported 6g networks," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 6171–6176.

[13] M. A. Enright, E. Hammad, and A. Dutta, "A learning-based zero-trust architecture for 6g and future networks," in *2022 IEEE Future Networks World Forum (FNWF)*, 2022, pp. 64–71.

[14] S. A. Chaudhry, A. Irshad, M. A. Khan, S. A. Khan, S. Nosheen, A. A. AlZubi, and Y. B. Zikria, "A lightweight authentication scheme for 6g-iot enabled maritime transport system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2401–2410, 2023.

[15] J. N. Khasnabish, M. F. Mithani, and S. Rao, "Tier-centric resource allocation in multi-tier cloud systems," *IEEE Transactions on Cloud Computing*, vol. 5, no. 3, pp. 576–589, 2017.
[16] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez, W. Wei, H. Song, A. Zomaya, and S. Khan, "System modelling and performance evaluation of a three-tier cloud of things," *Future Generation Computer Systems*, vol. 70, pp. 104–125, 2017.
[17] J. Islam, T. Kumar, I. Kovacevic, and E. Harjula, "Resource-aware dynamic service deployment for local iot edge computing: Healthcare use case," *IEEE Access*, vol. 9, pp. 115 868–115 884, 2021.
[18] Z. Zhai, K. Xiang, L. Zhao, B. Cheng, J. Qian, and J. Wu, "Iot-recsm—resource-constrained smart service migration framework for iot edge computing environment," *Sensors*, vol. 20, no. 8, p. 2294, 2020.
[19] R. Kumar, S. K. Gupta, H.-C. Wang, C. S. Kumari, and S. S. V. P. Korlam, "From efficiency to sustainability: Exploring the potential of 6g for a greener future," *Sustainability*, vol. 15, no. 23, p. 16387, 2023.
[20] V.-L. Nguyen, P.-C. Lin, B.-C. Cheng, R.-H. Hwang, and Y.-D. Lin, "Security and privacy for 6g: A survey on prospective technologies and challenges," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2384–2428, 2021.
[21] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of queueing theory*. John wiley & sons, 2011, vol. 627.
[22] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.
[23] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.
[24] J. D. Little and S. C. Graves, "Little's law," *Building intuition: insights from basic operations management models and principles*, pp. 81–100, 2008.
[25] N. U. Prabhu, *Foundations of queueing theory*. Springer Science & Business Media, 2012, vol. 7.
[26] M. Pinsky and S. Karlin, *An introduction to stochastic modeling*. Academic press, 2010.
[27] K. Haque, "Hajautettu pub-/sub-arkkitehti reaaliaikainen potilaan etävalvonta," Master's thesis, K. Haque, 2024.
[28] Y. Wang, J. Yang, Y. Chen, H. Liu, M. Gruteser, and R. P. Martin, "Tracking human queues using single-point signal monitoring," in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, 2014, pp. 42–54.
[29] D. Wu, H. Xu, Z. Jiang, W. Yu, X. Wei, and J. Lu, "Edgelstm: Towards deep and sequential edge computing for iot applications," *IEEE/ACM Transactions on Networking*, vol. 29, no. 4, pp. 1895–1908, 2021.
[30] H. Sedjelmaci and N. Ansari, "Zero trust architecture empowered attack detection framework to secure 6g edge computing," *IEEE Network*, vol. 38, no. 1, pp. 196–202, 2024.
[31] T. T. Huong, T. P. Bac, D. M. Long, B. D. Thang, N. T. Binh, T. D. Luong, and T. K. Phuc, "Lockedge: Low-complexity cyberattack detection in iot edge computing," *IEEE Access*, vol. 9, pp. 29 696–29 710, 2021.
[32] F. J. Dian and R. Vahidnia, "Formulation of ble throughput based on node and link parameters," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 4, pp. 261–272, 2020.
[33] F. J. Dian, A. Yousefi, and S. Lim, "A practical study on bluetooth low energy (ble) throughput," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2018, pp. 768–771.
[34] J. Siva and C. Poellabauer, "Connection-oriented ble traffic servicing characteristics on android devices," in *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, 2020, pp. 1–6.
[35] A. Zuiev, V. Krylova, A. Hapon, and S. Honcharov, "Research of microprocessor device and software for remote control of a robotic system," *Technology audit and production reserves*, vol. 1, no. 2/75, pp. 31–37, 2024.
[36] A. Maier, A. Sharp, and Y. Vagapov, "Comparative analysis and practical implementation of the esp32 microcontroller module for the internet of things," in *2017 Internet Technologies and Applications (ITA)*, 2017, pp. 143–148.
[37] A. Samefors and F. Sundman, "Investigating energy consumption and responsiveness of low power modes in micropython for stm32wb55," 2023.

Cinthya Celina Tamayo Gonzalez received the B.Sc. degree in telecommunications engineering from the National Autonomous University of Mexico, Mexico City, Mexico in 2020 and the M. Sc. degree in Cybersecurity from the University of Padova, Padua, Italy, in 2023. She is currently pursuing a PhD degree in the IMT School for Advanced Studies Lucca with the SySMA research group. Her research interests include lightweight security, hardware-based authentication and physical layer security.



Ijaz Ahmad is a doctoral researcher at the Centre for Wireless Communications – Networks and Systems (CWC-NS) research unit, Faculty of Information Technology and Electrical Engineering (ITEE), University of Oulu, Finland. He completed his M.Sc. degree in 2012 from the National University of Sciences and Technology, Pakistan. Currently, he is working on adaptive security in next-generations networks for digital healthcare, focusing on intelligent, trustworthy distributed computing and communication architectures. He has a background in cybersecurity, cryptography, and IoT communications.

Simone Soderi (SMIEEE) received his M.Sc. degree in 2002 from the University of Florence and the Dr.Sc. degree in 2016 from the University of Oulu, Finland. His expertise ranges from cybersecurity and wireless communications to embedded systems. He is currently an Assistant Professor at the IMT School for Advanced Studies Lucca, Italy, and an Adjunct Professor at the University of Padua, Italy, where he teaches in the master's degree program in cybersecurity. His research topics include cybersecurity for critical infrastructure systems, 6G, covert channels, network security, physical layer security, electromagnetic emission security, VLC, and UWB. He has been a TPC member of several conferences and a reviewer of many IEEE Transactions. He is the scientific leader of an industrial project investigating network security. Dr. Soderi has published journal and conference papers and book chapters. He holds five patents on wireless communications and vehicle positioning.

Erkki Harjula is a tenure-track assistant professor at the Centre for Wireless Communications – Networks and Systems (CWC-NS) research unit. He leads a team of six researchers, called Wireless system level architectures for future digital healthcare (WEALTH), focusing on communication architectures for healthcare use cases. The key technologies include Edge-cloud continuum, Edge-AI, Medical IoT, Network Security and Green communications. He received a D.Sc. degree in 2016, and a M.Sc. degree in 2007. Previously he has worked as a postdoctoral researcher and project manager at CWC-NS (2016-2020), and a research scientist / project manager for Center for Internet Excellence (CIE) (2013-2015), and for MediaTeam (2000-2014) research groups. He visited Columbia University in 2008-2009. Dr. Harjula is Principal Investigator in Eware-6G, Tomohead (Co-PI), Tech2Heal and Distech-6G projects, and he is the responsible teacher of Introduction to Internet graduate course. He is also an associate editor in Springer Wireless Networks (WINE) journal.