Journal of Information and Organizational Sciences
Volume 50, Number 2 (2026)
Journal homepage: jios.foi.hr

JIOS

DOI: 10.31341/jios.50.2.1
Open Access

UDC 004.738.5:004.8
Original Scientific Paper

QoS-Aware Decision-Making Framework for Service Allocation in IoT Architectures

Alem Čolaković[1*], Bakir Karahodža[1], Samir Čaušević[1] and Jasmina Baraković Husić[2]

[1]University of Sarajevo, Faculty of Traffic and Communications, Sarajevo, Bosnia and Herzegovina
[2]University of Sarajevo, Faculty of Electrical Engineering, Sarajevo, Bosnia and Herzegovina
[*]Correspondence: alem.colakovic@fsk.unsa.ba

PAPER INFO

Paper history:
Received 07 March 2026
Accepted 23 July 2026

Citation:
Čolaković, A., Karahodža, B., Čaušević S. & Baraković Husić, J. (2026). QoS-Aware Decision-Making Framework for Service Allocation in IoT Architectures. In Journal of Information and Organizational Sciences, vol. 50, no. 2, pp. 263-292

ABSTRACT

The increasing complexity of Internet of Things (IoT) environments requires adaptive mechanisms for efficient service allocation across distributed infrastructures. Existing approaches are often focused on specific optimization algorithms or isolated Quality of Service (QoS) parameters, lacking a unified framework for decision-making across IoT, Fog/Edge, and Cloud layers. This paper proposes a modular QoS-aware decision-making framework that integrates QoS profiles, key performance indicators (KPIs), utility functions, and multi-criteria decision-making mechanisms to support both static and dynamic service allocation. The framework considers service execution latency, energy consumption, network throughput, and network coverage as decision criteria and enables adaptive balancing of conflicting QoS requirements. Its applicability is demonstrated through a smart transportation use case involving multiple service allocation scenarios across CRU, Fog, and Cloud infrastructures. The results confirm that different allocation strategies exhibit distinct QoS trade-offs and demonstrate the suitability of the proposed framework for adaptive and resource-efficient service allocation in layered IoT architectures.

Keywords: Internet of Things (IoT), Quality of services (QoS), Resource management, QoS-aware mechanism, Edge-Fog-Cloud computing, IoT Architecture

1. Introduction

The development of the Internet of Things (IoT) represents one of the key technological transformations of modern society, enabling the interconnection of a vast number of heterogeneous devices for the purpose of data collection, processing, and exchange. IoT systems are utilized across a wide spectrum of applications—from smart homes and cities, through transportation and logistics, to industrial automation and healthcare. Despite their growing adoption, a fundamental challenge remains the limitation of hardware and software resources, which complicates the execution of complex data processing and storage tasks.

To overcome these limitations, IoT is increasingly being integrated with both distributed and centralized computing paradigms, among which Edge, Fog, and Cloud systems stand out. The integration of IoT architecture with various computing models (Cloud, Fog, MCC, MEC, Edge) directly affects system performance, including data processing and transmission time, reliability, energy requirements, and security (Abdulazeez & Askar, 2023; Čolaković, 2023; Firouzi et al., 2022). Cloud infrastructure provides virtually unlimited resources but introduces issues of latency, dependency on internet connectivity, and data transfer costs. On the other hand, Edge and Fog approaches reduce network latency and congestion but offer more limited capacities for complex analytics and long-term data storage. Due to these contrasting characteristics,

selecting the optimal integration model and service allocation strategy becomes a complex optimization problem.

Previous studies highlight the need for efficient resource management strategies to achieve the desired level of QoS through task allocation to the appropriate infrastructure (Alonso et al., 2020; Čolaković, 2023; Zahoor & Mir, 2021). Key parameters such as latency, energy consumption, reliability, and cost are most commonly expressed through QoS metrics, which enable the quantitative evaluation of alternative solutions. Various approaches have been proposed in the literature: the development of a "smart middleware layer" between IoT devices and infrastructure to optimize resources (Tuli et al., 2019; Xavier et al., 2020; Zhang et al., 2020); formal approaches based on rewriting-based techniques for infrastructure reconfiguration (Haidri et al., 2020; Khebbeb et al., 2020); data caching to reduce latency (Safavat et al., 2020; Wei et al., 2020); algorithmic solutions such as PSO (Gill et al., 2019); the improved firefly algorithm (Bacanin et al., 2022); deep learning (DRL, DDQN-PER) (Sharma & Thangaraj, 2024), heuristic/dynamic algorithms ("IoT Application Modules Placement and Dynamic Task Processing in Edge-Cloud Computing," 2020); metaheuristics (NSGA-II, GWO) (Kaushik & Hamed, 2022); SDN and adaptive protocols (Fröhlich et al., 2021); genetic algorithm for energy efficient fog layer resource management (Reddy et al., 2020), and distributed deep reinforcement learning approaches (S. Wang et al., 2023). Although these studies provide significant contributions to the modeling and optimization of IoT systems, most solutions rely on specific algorithms and face challenges such as device heterogeneity, the need for energy efficiency, real-time processing, and system scalability (Bu et al., 2023; Maray & Shuja, 2022; Sinha Roy et al., 2019; Zhou et al., 2023). Additionally, the limitations include high computational complexity, simplified assumptions, and a lack of validation in real IoT scenarios (Gasmi et al., 2022).

However, in practice, numerous challenges arise in the integration and distribution of services across complex IoT architectures, particularly when multiple metrics and KPIs (Key Performance Indicators) are taken into account. The reviewed literature reveals a lack of proposals for an integrated approach that would synergistically encompass the creation of QoS profiles based on application requirements, the evaluation of system components both individually and in integration, the observation of different technologies' behavior across various models, and the development of practical and resource-efficient solutions. In this context, the main objective of this paper is to propose a QoS-aware architecture and strategic framework for service allocation that enable multi-criteria decision-making and practical applicability across diverse IoT scenarios.

Unlike previous studies focused on individual algorithmic optimizations, this paper offers a formalization of the service allocation process through a QoS-aware architecture that integrates both static and dynamic decision-making approaches, independent of specific algorithmic implementations. Based on this, the following key research question is posed: How can an adaptive and resource-efficient mechanism be developed for optimizing service allocation across IoT architecture layers, ensuring practical applicability across different architectural styles? In line with this question, the main objective of the paper is to develop a strategic framework for service and resource allocation across multiple layers of the system architecture and to propose a decision-making mechanism that integrates static and dynamic approaches. In this way, the paper contributes to the development of intelligent and adaptive IoT systems capable of meeting heterogeneous user and application requirements, while also providing a foundation for further research in the field of QoS performance optimization in distributed environments.

The remainder of this paper is organized as follows. Section 2 presents the conceptual architecture of the layered IoT system and discusses the integration of IoT devices with Edge/Fog and Cloud infrastructures. Section 3 formalizes the service allocation problem within layered IoT architectures. Section 4 defines the key performance indicators (KPIs), QoS profiles, utility functions, and multi-criteria decision-making mechanisms used to support service allocation decisions. Section 5 presents the modular QoS-aware decision-making mechanism and discusses alternative architectural implementation styles. Section 6 provides a scenario-based evaluation of the proposed framework through a smart transportation use case, including KPI analysis and utility-based assessment of alternative service allocation strategies. Finally, Section 7 concludes the paper and outlines directions for future research.

2. Layered architecture of IoT systems

IoT devices are limited in terms of memory and processing capabilities and therefore rely on the support of computing systems such as Edge, Fog, and Cloud infrastructures. Cloud computing offers virtually unlimited resources for data storage and processing but introduces challenges such as dependency on internet availability, increased latency, and data transfer costs. In response, concepts such as Fog, Edge, MEC, MCC, and Cloudlet have been developed to enable service execution closer to the data source, thereby reducing latency, increasing reliability, and decreasing network load. However, local infrastructure cannot match the

processing capacity of the cloud, which complicates the execution of complex analytics, large-scale user access, or long-term data storage (Babovic et al., 2016). This highlights the importance of selecting an appropriate system architecture and integration model for IoT with other computing systems.

Essentially, these concepts form an integral part of modern IoT architecture, which is most commonly modeled as a multilayer structure - for example, three-layer (Ammar et al., 2018; Čolaković, 2023; Khattak & Khan, 2024), five-layer (Al-Fuqaha et al., 2015; Omoniwa et al., 2019), seven-layer (Al-Awami et al., 2023), and others. For the purposes of this paper, a three-layer architecture is used, which includes the perception layer (IoT devices), the Edge/Fog layer, and the Cloud layer. All layers can communicate and cooperate with one another, and the choice of the integration model becomes a key factor in optimizing system performance. Therefore, the generic architecture of an IoT system (designed from the perspective of infrastructure, technologies, and functionalities) can be represented as a three-layer structure that incorporates different models of interaction between devices and computing systems for resource and task allocation across various layers. Thus, within the system architecture, three main layers can be identified: IoT devices (perception layer), the Edge/Fog layer, and the Cloud layer (Figure 1).

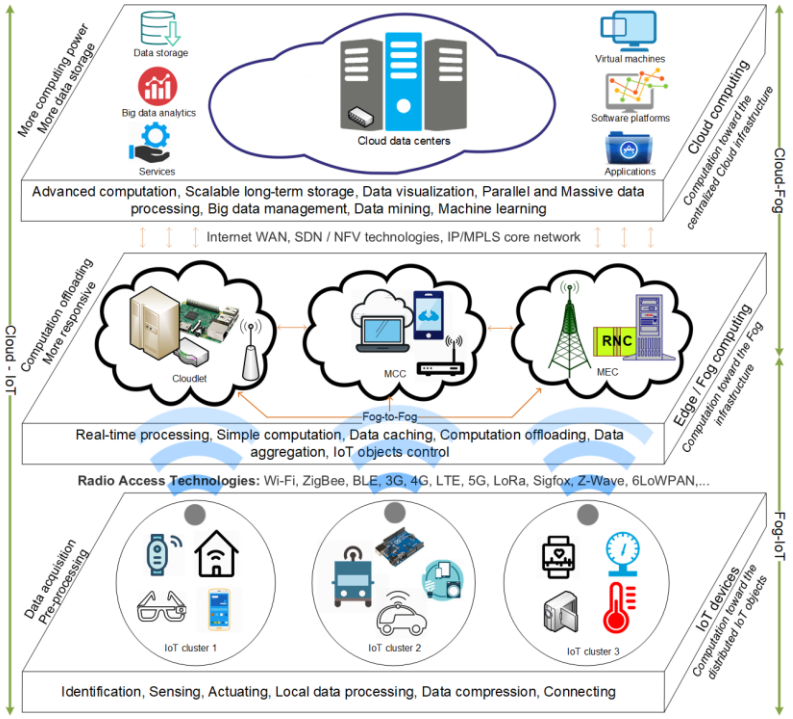

Figure 1. Layered architecture of IoT systems

Possible architectures and integration models offer various advantages but also certain limitations, which is why they need to be thoroughly analyzed in relation to expected performance and specific application contexts. The key challenges in selecting an appropriate architecture include achieving the desired performance (QoS), ensuring sufficient capacity for storing and processing large volumes of data, maintaining system reliability, supporting user and device mobility, and safeguarding data security and privacy (Chiang & Zhang, 2016; Madsen et al., 2013).

The performance of IoT systems largely depends on the data processing infrastructure, network characteristics, applied technologies, and the specific requirements of the application itself. Figure 2 illustrates a simplified workflow of service allocation based on the proposed IoT system architecture.

Cloud environments provide significantly higher processing and memory capacities, enabling rapid data processing but simultaneously introducing additional network latency due to the transfer of raw data over the Internet. In contrast, infrastructure located closer to the data source (Edge/Fog) reduces latency by eliminating the WAN segment but has limited processing capabilities, which slows down more complex computations. This contradiction clearly highlights the complexity of selecting the optimal integration model and the need for optimizing the utilization of available resources.

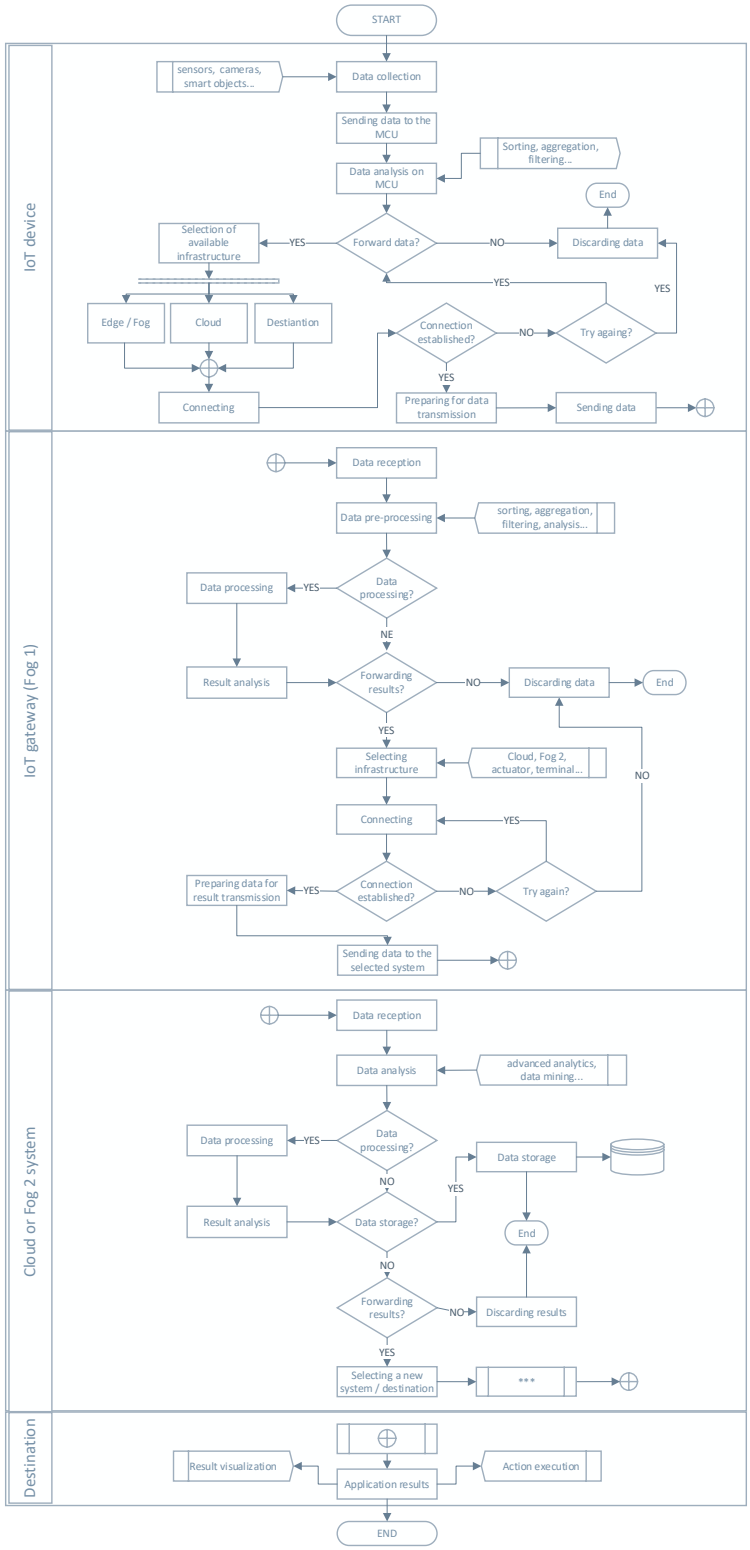

Figure 2. Service allocation processes across different systems for offloading IoT devices

IoT architectures based on integration with Cloud, Fog, and Edge systems raise issues related to various aspects such as performance, cost-efficiency, administrative challenges, legal and regulatory compliance, provider responsibility, privacy, and data protection. Data transmission from IoT devices to the Cloud introduces additional network latency due to Internet-based transfer. When the computing infrastructure is located closer to the data source (IoT devices), partial elimination of network latency is achieved. However, in that case, data processing becomes considerably slower due to the lower processing capabilities compared to those of Cloud infrastructures. This fact emphasizes the complexity of selecting an appropriate integration model and the challenge of optimizing the utilization of available resources. The general characteristics of different types of systems (Edge, Fog, MCC, MEC, Cloud)—according to the key parameters influencing QoS performance, on which service allocation from IoT systems can be performed—have been presented in previous research (Čolaković et al., 2025).

3. IoT service distribution model

Based on the description of the conceptual IoT system architecture, it is possible to propose and design feasible architectures, technologies, protocols, and infrastructure capable of meeting different QoS performance targets during their integration and the allocation of IoT services. Since the number of such solutions is virtually unlimited, it is necessary to use appropriate metrics. Performance metrics for evaluating IoT systems can be considered across different architectural layers, including applications, devices, the network, and cloud systems. They encompass parameters such as latency, reliability, security, energy consumption, resource utilization, and scalability, thereby enabling a comprehensive insight into Quality of Service (QoS). Therefore, it is necessary to capture and interpret the most relevant system performance parameters in a meaningful manner. However, due to the complexity and heterogeneity of IoT environments, it is practically impossible to simultaneously incorporate all possible QoS-related parameters into a generalized allocation model. Therefore, realistic assumptions were adopted to limit the number of analyzed parameters to a representative set of Key Performance Indicators (KPIs) capable of providing sufficiently reliable performance evaluation and decision-making support. In this context, identifying the most relevant KPIs and determining their relative importance for different IoT applications remain important open research challenges in the implementation of QoS-aware allocation mechanisms. Various multi-criteria decision-making (MCDM) methods, such as TOPSIS, AHP (Analytic Hierarchy Process), FAHP (Fuzzy AHP), and similar approaches, may be used to determine the significance and weighting factors of individual QoS indicators depending on the requirements and priorities of specific IoT application scenarios.

For the purposes of this paper, four metrics have been selected that are often regarded as key performance indicators of IoT systems. Service execution time (latency) directly affects the user experience and the ability of an IoT system to respond to requests in real time, which is particularly important for latency-sensitive applications (Fröhlich et al., 2021; Sharma & Thangaraj, 2024). Energy efficiency is crucial due to the limited battery capacity and resources of IoT devices, and optimizing energy consumption extends device lifespan and reduces operational costs (Almudayni et al., 2025; Hu et al., 2022; Sharma & Thangaraj, 2024). Also, network throughput is an important indicator of communication performance because it determines the effective capacity available for data transmission between IoT devices and processing infrastructure, directly affecting scalability and service responsivenes. Furthermore, network coverage represents a key deployment-related metric that reflects communication availability and reliability, ensuring continuous connectivity and uninterrupted service operation, particularly in mobile and geographically distributed IoT environments. Although additional QoS parameters such as reliability, security, scalability, throughput, and availability may significantly influence IoT service allocation, this work primarily focuses on latency and energy consumption as representative and widely adopted KPIs in IoT offloading and resource allocation scenarios. These parameters were selected because they directly affect real-time responsiveness, device autonomy, and resource efficiency, which are among the most critical constraints in layered IoT environments. Nevertheless, the proposed framework is not restricted to these indicators and can be extended to incorporate additional QoS metrics depending on the requirements of specific IoT applications and deployment scenarios.

Therefore, these metrics can demonstrate the efficiency of the proposed alternative based on the target values established during the optimization process:

- Total service execution time of the IoT service – T_{exe}
- Total energy consumed by an IoT device during service execution – E_{dev}
- Effective network throughput – Th
- Network coverage availability – Cov

When considering the T_{exe}, the optimization goal is defined by the condition that this time must be less than the requirement of the IoT application (T_{app}). Two approaches can be used for allocating services to available resources in order to achieve this goal: partial or complete distribution of specific services across the available resources. This can be formally expressed as follows (1):

$$T_{exe} = \alpha_0 \cdot t_L + arg \max_{i=1\ldots k}\{\alpha_i \cdot t_{F_i}\} + arg \max_{\mathrm{j=1\ldots c}}\{\alpha_j \cdot t_{C_j}\} + \sum_{\mathrm{i=1}}^{N}(\beta_i \cdot T_{cm_i}) \leq T_{app} \quad (1)$$

According to this objective, it is necessary to determine where specific services will be allocated and what percentage of processing will be executed on the device (α_0), while the remaining portions of the tasks will be allocated to offloading systems, including the Fog infrastructure (α_i) and the Cloud (α_j). In the case of multiple systems within the same architectural layer (i Fog system i j Cloud servers), the processing time for that layer is estimated based on the maximum time required by any system within the same layer to complete the assigned tasks. This approach assumes that the data being processed have simultaneously reached all systems within the same layer. This assumption represents a simplified approximation intended for the conceptual formulation of the proposed framework and may be applicable in scenarios where Edge/Fog nodes are deployed within relatively close network proximity and interconnected through high-speed communication links. In heterogeneous IoT environments, transfer times may significantly differ due to varying network conditions, node distribution, bandwidth limitations, and communication overhead. However, the primary objective of this work is not detailed network-aware latency modeling, but the development of a generalized QoS-aware decision-making framework for service allocation across layered IoT architectures. Therefore, the proposed model should be interpreted as a generalized abstraction intended to support the formalization of allocation mechanisms, while more detailed network-aware modeling and latency estimation can be incorporated in future implementations and experimental evaluations. However, if certain processes are allocated to offloading computing systems, the total time must also include the time required to transfer a specific portion of data (β_i) to these systems (across different layers of the system architecture), where the data traverse N network segments.

The allocation parameters α_i and β_i do not represent fixed coefficients, but dynamic allocation variables determined according to current QoS requirements, available computational resources, network conditions, and application priorities. These variables may be estimated using profiling, benchmarking, simulation-based evaluation, historical monitoring data, or adaptive optimization mechanisms integrated within the QoS decision-making module. In static allocation approaches, predefined values can be derived from simulation or experimental measurements under known operating conditions, whereas in dynamic environments the values may be continuously adjusted according to real-time KPI measurements, system load, and resource availability.

The second key indicator concerns the power supply of IoT devices and their energy consumption; therefore, by analogy with the previous expression, the following problem can be formulated (2):

$$E_{dev} = \alpha_0(P_L \cdot t_L) + P_{id} \cdot arg \max_{\mathrm{j=1\ldots k}}\{\alpha_j \cdot t_{off_j}\} + P_{tx} \cdot \sum_{\mathrm{i=1}}^{n}(\beta_i \cdot t_{tx_i}) \leq E_{app} \quad (2)$$

In the case of energy consumption, E_{dev} represents the total energy consumed by an IoT device during the execution of all required services and their associated computational and communication processes. The optimization goal is that this value does not exceed the energy constraint defined by the application requirements (E_{app}).P_L denotes the power required for local processing, i.e., executing services directly on the IoT device; P_{id} represents the power consumption in the device's idle state; while P_{tx} refers to the power required for transmitting data over the network to n systems used for offloading IoT device tasks.

The previously defined tasks can be reduced to an optimization problem of finding the values for α_0, α_j, and β_i within any of the given requirements, depending on the number of subsystems engaged for device offloading. The ultimate goal is to allocate services across different system components to achieve minimal energy consumption.

The optimization objective associated with network throughput is to maximize the effective communication capacity available for data transmission between the CRU, Fog/Edge, and Cloud layers. Higher throughput values enable faster transmission of large volumes of IoT data, improve scalability, and reduce the risk of communication bottlenecks under increasing traffic loads.

Network coverage represents the availability and continuity of communication connectivity within the considered operational environment. Since uninterrupted communication is particularly important for mobile and safety-critical IoT applications, the optimization objective is to maximize communication coverage and service availability along the transportation corridor.

To derive an appropriate model for the integration and distribution of IoT services, it is necessary to adhere to the previously described parameters that affect QoS performance and the defined optimization

tasks. In addition to these goals and constraints, the specifics of the multilayer IoT architecture and limitations must be considered, including adherence to the principles of simplicity, interoperability, accuracy, and acceptability in real-world environments. Creating such a model requires a formal strategic framework that should address several questions:

- Which services and associated resources will be allocated outside the IoT device?
- Under which requirements will the IoT device be offloaded?
- Which mechanism will be used for the allocation process?
- To which layers of the system architecture will services and resources be distributed?
- How will the distribution of services be monitored?

Based on these questions, a formal framework is proposed that provides a detailed description of the necessary activities and corresponding methodologies for the IoT service distribution model. It consists of five steps that answer the stated questions, along with a description of the main activities and the desired outcomes. The Figure 3 graphically illustrates the steps along with the main activities and corresponding outcomes. As stated in Step 3, the design of the allocation approach, two fundamental approaches can be used for allocating services and resources across different layers of the system architecture:

- Static decision-making: if the allocation is based on preconfigured instructions at each layer.
- Dynamic decision-making: if the decision-making mechanism relies on the current state of resources within the entire system and the real-time requirements of individual applications.

Static decision-making is implemented in systems with limited resources and in cases where the expected system load can be estimated in advance. The estimation is performed based on simulations of real environments and laboratory conditions. To implement dynamic decision-making, higher system performance must be ensured. The data processing required for decision-making is typically deployed on network gateways or dedicated infrastructure in the case of larger systems. Therefore, static models can be efficient in predictable and stable environments, while dynamic approaches offer advantages under varying load conditions and heterogeneous application requirements.

Step 1: Identification of services and available resources

Main activities: Identification and classification of all tasks on the IoT device side based on criteria and requirements for total data processing time, data transmission time, and energy consumption:

- Listing all IoT device tasks with classification of individual functions for data acquisition (sensing), processing, and control.
- Determining metrics for each task: expected execution time, data volume, and energy footprint. It is also necessary to determine the criticality of task execution.
- Grouping tasks into categories according to execution time: real-time control, periodic monitoring, serial or batch analytics, etc.
- Defining resource requirements (e.g., CPU cycles, memory, network I/O) for the purpose of creating service and resource profiles needed for decision-making.

Outcome: Taxonomy of services and device profiles

Step 2: Definition of allocation criteria

Main activities: Establishing measurement thresholds for different QoS parameter values:

- Setting latency thresholds aligned with application requirements, e.g., below 100 ms for control loop operations.
- Defining the total amount of energy per device that triggers a specific allocation mechanism, e.g., initiating remote processing when local energy consumption reaches a predefined limit.
- In addition to these basic criteria related to T_{exe} and E_{dev}, other criteria are also established:
- Including network state variables such as bandwidth utilization, packet loss, and connection reliability.
- Developing context-aware rules (time of day, geographic location, device mobility) that enable or disable offloading.

Outcome: Set of offloading rules

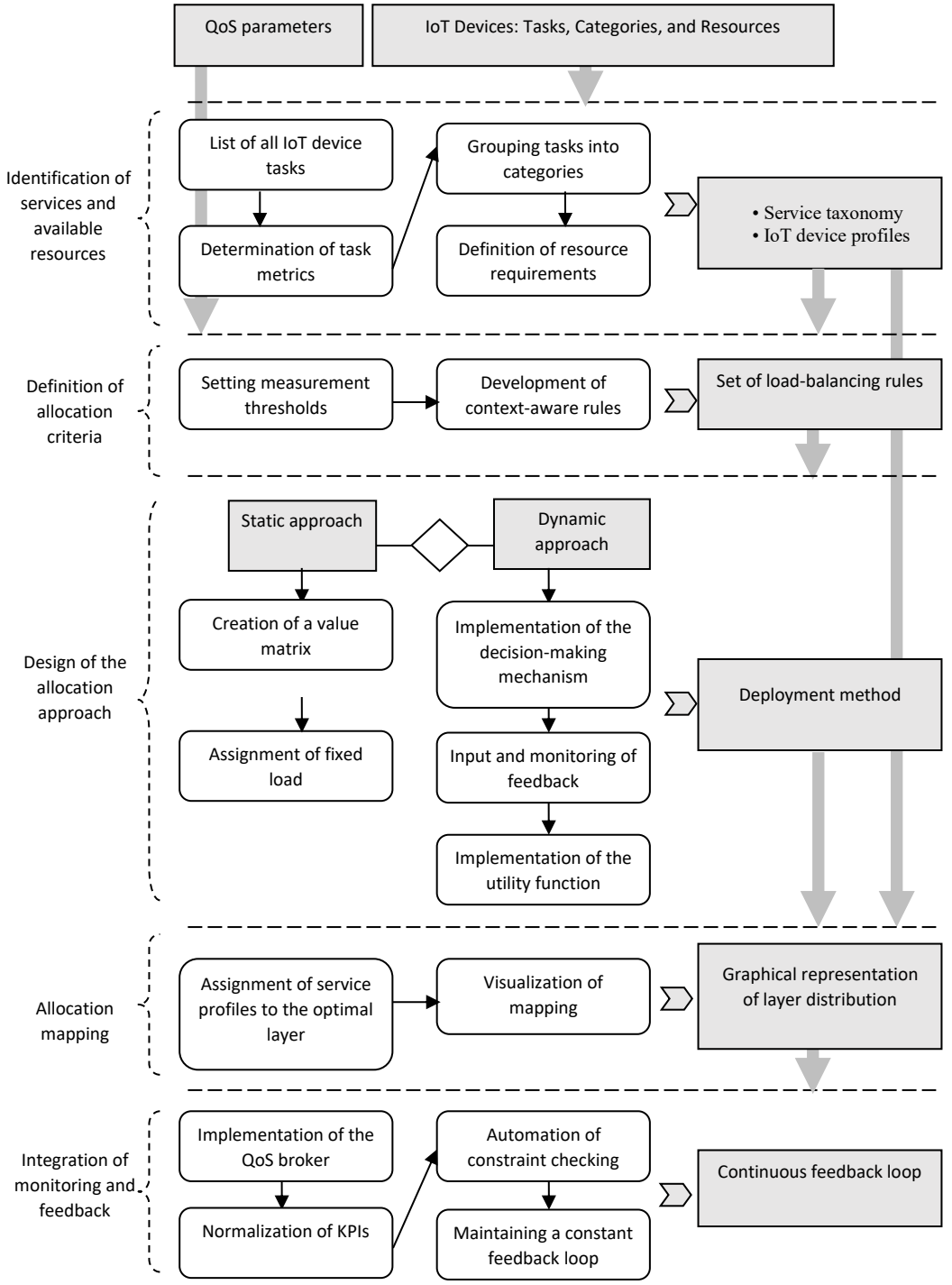

Figure 3. Service allocation flow in a layered IoT architecture

Step 3: Design of the allocation approach
Main activities: It is necessary to determine the choice between a static and a dynamic approach to allocation (explained in detail below the table). In the case of a static approach, it is necessary to:

- Create value matrices suitable for lookups, based on testing under known traffic and load conditions.
- Assign a fixed load ratio for tasks among individual IoT devices, Edge/Fog/Cloudlet devices, and the Cloud infrastructure.
- In the case of a dynamic approach, it is necessary to:
- Implement a decision-making mechanism, e.g., an ML classifier or a rule-based system on gateways or brokers.
- Continuously input and monitor feedback on QoS parameters and KPIs in order to support real-time task redistribution.
- Implement utility functions to balance multiple objectives (e.g., latency (time) vs. energy vs. cost).

Outcome: Selection of deployment method

Step 4: Allocation mapping
Main activities: Mapping each service to the layers (perception, Edge/Fog, Cloud) based on profiles and the defined offloading criteria. A layered QoS model is used:

- Perception layer (device), which filters and performs initial processing of critical/raw data.
- Edge/Fog/Cloudlet, which executes latency-sensitive tasks and/or tasks that require fewer computational resources.
- Cloud layer, where computationally demanding analytics and long-term data storage are performed.
- To perform this mapping, it is necessary to:
- Assign each defined service profile to its optimal layer based on the principle of trade-offs, using requirements for computation, memory, and network load.
- Visualize the mapping using a suitable tool by means of an allocation diagram or a service distribution matrix, with the goal of simplifying implementation.

Outcome: Graphical representation of layer-based distribution

Step 5: Monitoring, evaluation, and feedback
Main activities: It includes the implementation of a QoS broker and manager for collecting KPIs, normalizing metrics, as well as dynamically or statically adjusting the allocation. The required steps are:

- Implement a QoS broker at each layer to collect performance metrics: throughput, jitter, CPU load.
- Normalize KPIs and forward them to a central QoS manager for cross-layer analysis.
- Automate the verification of constraints and defined thresholds. Example: if a KPI falls outside acceptable bounds, activate relocation rules.
- Maintain a continuous feedback loop for optimization.

Outcome: Continuous feedback loop

4. Multi-criteria decision-making

Previously, two formalized approaches were presented for allocating services to the appropriate available resources to achieve the objectives defined in (1) and (2), each designed for optimization with respect to a single indicator –T_{exe} ili E_{dev}. However, the problem becomes more complex when multiple criteria must be considered simultaneously, for example, when optimization must account for both service execution time and energy consumption. Such a requirement calls for a method that incorporates multiple metrics in order to find an acceptable implementation for each layer of the system architecture. In this case, achieving the desired performance level of individual IoT applications is viewed as the cumulative effect of the service, network, and perception layers on the application (Figure 4).

The need for such a method was previously identified as one of the activities within the strategic framework in Step 3, *Design of the allocation approach*, which is used when selecting the dynamic allocation approach. It is implemented in the form of a utility function to balance multiple objectives or criteria, such as time, energy, and/or cost.

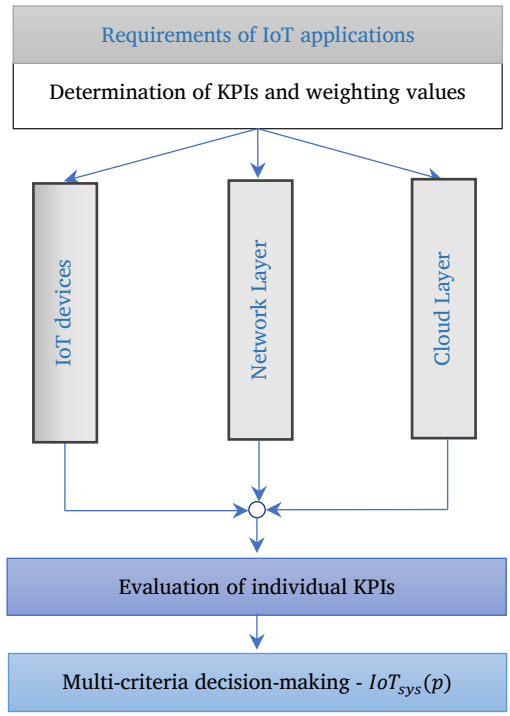

Figure 4. Concept of IoT system evaluation

In a previous study (Čolaković, 2023) a multi-criteria decision-making method was proposed that can be used as an effective tool for applying various metrics within a normalized range, independent of their measurement units. This approach is essential for analyzing a system while taking into account different KPIs across all layers of the system architecture. In this context, utility functions, $f(p_i)$, were used to evaluate or analyze the IoT system IoT_{sys} based on various key metrics (p_i). These mathematical functions enable the ranking of alternatives according to their utility for specific application cases, where the overall system performance level is equal to the sum of the obtained functions for individual KPIs (with n representing the number of key parameters), each multiplied by its corresponding weighting factor w_i (3):

$$IoT_{sys}(p) = \sum_{i=1}^{n} w_i \cdot f(p_i) \tag{3}$$

Individual utility functions $f(p_i)$ correspond to specific value functions used to evaluate each KPI. All analyzed parameters should be scaled within the same range, e.g., [0,1]. The specific value functions are multiplied by weighting coefficients to account for the fact that some KPIs are more important than others. Since $IoT_{sys}(p)$ represents the level of system or application performance depending on the subject of analysis, the problem formulation can be defined as a decision-making process for the distribution (allocation) of IoT services and resources across different layers of the system architecture, with the aim of improving performance and optimizing the utilization of available system resources.

The weighting factors w_i represent the relative importance of each individual key indicator (i) for a given QoS profile of an application and are determined according to the requirements of IoT applications. Various methods can be used to determine these weights, such as the AHP or FAHP, among others. The individual utility functions $f(p_i)$ correspond to specific value functions used to evaluate each KPI.

Determining $f(p_i)$ for each KPI is a complex problem that, in the referenced study, was addressed by adapting so-called *sigmoid functions*. Two different cases may occur: $f(p_i) = \{f(p_i \uparrow); f(p_i \downarrow)\}$, depending on the meaning of the analyzed parameters. For example, for KPIs such as network throughput or resource utilization, a higher value represents better performance. Therefore, a higher value of the utility function for such parameters—i.e., a higher value of the sigmoid function $f(p_i \uparrow)$—indicates improved system performance (4). Conversely, some other parameters, such as execution time and energy consumption, must be minimized to increase performance levels; in such cases, the corresponding utility functions should reduce the value of the sigmoid function $f(p_i \downarrow)$ in order to achieve better performance (5).

$$f(p_i \uparrow) = L\left(\frac{A}{1 + e^{-\alpha_k \frac{(p_i - r_i)}{Z_i}}} - U\right) \tag{4}$$

$$f(p_i \downarrow) = 1 - L\left(\frac{A}{1 + e^{-\alpha_k \frac{(p_i - r_i)}{z_i}}} - U\right) \tag{5}$$

In the given formulas, A represents the maximum possible value of the function $f(p_i)$. If the real values of all functions are assumed to be within the range [0,1] to normalize the values of different parameters, then the coefficient value is $A = 1$. The parameter p_i denotes the measured (or estimated) performance level for a specific KPI.

The slope of the function depends on the parameters α_k and z_i, which are adjustable parameters used to tune the dimensionless values of the functions. By increasing or decreasing these parameters, the "sensitivity" of an application to individual performance factors can be adjusted. For example, in the case of $f(p_i \uparrow)$, an IoT application remains relatively stable in performance (i.e., less sensitive) for smaller values of α_k and larger values of z_i for certain KPIs.

The parameter r_i represents sensitivity to performance degradation and defines the midpoint (central point) of the sigmoid function curve. This parameter can be interpreted as a reference value for the KPI: a higher value indicates a more demanding application in the case of $f(p_i \uparrow)$, while for $f(p_i \downarrow)$, a lower value of this parameter indicates greater sensitivity to that particular metric.

The values L and U are introduced to ensure that the utility function attains appropriate values in boundary cases, i.e., $(0) = 0$ and $f(\infty) = 1$, indicating that the unavailability of certain resources results in a zero performance level, while an "infinite" allocation of resources leads to the maximum performance level (6).

$$L = \frac{1 + e^{\frac{\alpha_k r_i}{z_i}}}{e^{\frac{\alpha_k r_i}{z_i}}}, \; U = \frac{1}{1 + e^{\frac{\alpha_k r_i}{z_i}}} \tag{6}$$

The reference values (r_i) may represent application requirements for a specific KPI ($p_{i_{req}}$) or a recommended value for a given parameter. For some parameters, these values can be obtained from established recommendations (e.g., network QoS parameter guidelines). However, for most IoT system performance indicators, existing recommendations cannot be directly applied, as previously discussed in this paper. Another way to obtain these values is to use the maximum or average measured (or estimated) value for each parameter p_i. One of the key advantages of applying the proposed and adapted sigmoid functions for utility computation is their flexibility in determining reference values.

In accordance with the previous considerations, the utility functions have the following main properties:

- $IoT_{sys}(p)$ is an increasing function in the case of improved system performance.
- $f(p_i)$ is an increasing function in the case of improvement of an individual parameter p_i.
- $IoT_{sys}(p)$ and $f(p_i)$ are scaled within the interval [0,1].

A higher value indicates a positive effect on performance, and the objective formulation can be expressed as follows (7):

$$\max_{p_i} IoT_{sys}(p) \tag{7}$$

The described method, which was validated in a previous study (Čolaković, 2023), offers several significant advantages, such as independence from specific KPIs, the possibility of easier selection of reference values, independence from measurement units and value ranges, and the ability to adjust desired outcomes for clearer visualization of the utility function through the tuning of application sensitivity parameters, among others.

The QoS-aware IoT architecture proposed in this paper, together with the method described in the aforementioned study, can serve as a framework for developing a decision-making module responsible for distributing (allocating) tasks across available resources. For this purpose, the proposed QoS architecture should be applied to decision-making in accordance with the corresponding IoT application profile. However, future research should address the open question of determining the appropriate set of key indicators for each application and their weighted values, which should be mapped across all layers of the system architecture.

5. QoS-aware mechanism for IoT service allocation

To select the best option among possible scenarios involving different offloading models and technologies, a matrix is constructed for choosing the optimal IoT system model (8). For each possible scenario, it is necessary to calculate the value of the utility function $IoT_{sys}(p_{kn})$ depending on the different system loads S_{w_k} and the technologies applied within various integration models IM_n.

$$IoT_{sys}(C) = \left\{ \begin{array}{c} \overbrace{\begin{matrix} System\ Load \\ S_{w_1} \\ \vdots \\ S_{w_k} \end{matrix}}^{} \quad \overbrace{\begin{matrix} Offloading\ Models \\ IM_1 \quad \cdots \quad IM_n \\ IoT_{sys}(p_{11}) \quad \cdots \quad IoT_{sys}(p_{1n}) \\ \vdots \quad \ddots \quad \vdots \\ IoT_{sys}(p_{1k}) \quad \cdots \quad IoT_{sys}(p_{kn}) \end{matrix}}^{} \\ \underbrace{IoT_{sys}(p) = \langle p_{i_{req}}, p_i \rangle, \quad i - \text{no. of } KPIs}_{} \end{array} \right\} \tag{8}$$

In the presented matrix, k represents the number of different system loads, which may include communication and computational processes. For example, the amount of data generated and entering the system. These system loads depend on various factors, such as the number of IoT devices and the rate at which they collect and transmit data.

By applying the proposed matrix and analyzing the key indicators, the optimal system configuration – $IoT_{sys}(C)$ —is obtained as a function of different loads and available integration models. The described procedure is used in the creation of decision-making mechanisms implemented on systems such as devices or network gateways. As mentioned in the previous section, these mechanisms can be implemented either as statically predefined, based on pre-established criteria, or as dynamic, relying on continuous monitoring of system performance.

The proposed mechanism consists of several modules (Figure 5) and is implemented with a configuration that supports both static and dynamic decision-making. In the case of dynamic decision-making, appropriate decision-making and optimization mechanisms may be employed to continuously monitor system state changes and identify recurring patterns. By understanding system capabilities and available resources, it becomes possible to estimate or measure performance under different conditions and support service allocation decisions across the layered IoT architecture. Future implementations of the framework may additionally incorporate machine learning techniques to enhance prediction and adaptive decision-making capabilities.

Each module serves a specific purpose within the decision-making mechanism, contributing to the process of determining service allocation. The individual components of the proposed mechanism are explained in detail below.

The mechanism for deciding on service allocation can be developed as part of the program code, whose operating algorithm can be represented using a workflow diagram (Figure 6).

The **Context Module** is a component that enables differentiation of the conditions under which a specific system or its components operate. This involves awareness of the system's available resources (e.g., CPU, RAM, memory)— $SYS(C)$ (9) - which refer to the characteristics and configurations of all computing components, including IoT devices $D_{sp}(C)$, offloading systems (Fog and Cloud infrastructure) $O_{sp}(C)$, and network settings $N_{sp}(C)$.

$$SYS(C) = \{D_{sp}(C), N_{sp}(C), O_{sp}(C)\} \tag{9}$$

It is necessary to understand the connectivity capabilities, i.e., the integration models or profiles $N_{mod}(C)$, to determine the $KPI(p_i, w_i)$,, the system's QoS requirements $QoS(r)$, and to monitor its load $S_w = \langle D_{rate}, t_{w_s} \rangle$. Based on these indicators, a *CP (Context Profil)* is created - a database of possible states in which the system may operate (10).

$$CP = \{SYS(C), KPI, QoS(r), S_w\} \tag{10}$$

The **Evaluation Module** evaluates the performance of the selected KPIs depending on the load and available resources, as well as the weighted values of individual indicators. In the case of static decision-making, a matrix is defined for selecting optimal solutions based on testing, whereas in dynamic decision-making, components for automatic performance assessment are implemented. Performance assessment entails updating the matrix, since the operating conditions of the system may change (e.g., changes in system load). The **Decision Module** determines the best alternative to achieve the maximum value of the utility function based on the matrix for selecting the optimal solution. In addition, when a decision is made to offload the IoT device, the availability of resources and services at that moment is verified and a connection to another system is executed. In the event of errors, an alternative solution can be selected, with record-keeping so that, in the case of recurring errors, the necessary procedures can be undertaken.

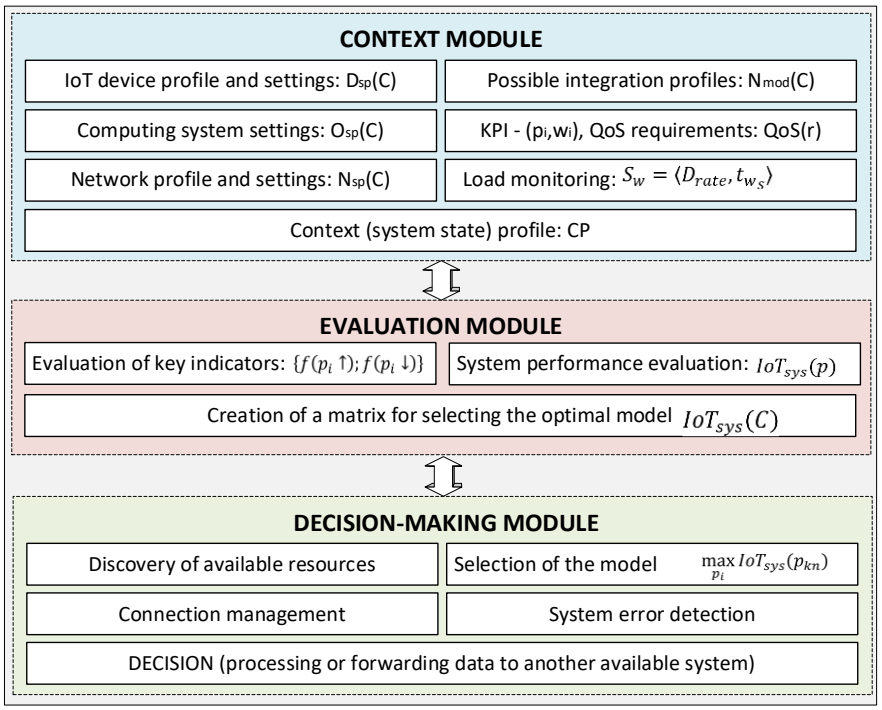

Figure 5. Framework and main components of the service allocation decision-making mechanism

The proposed formulation is intended to provide a generalized QoS-aware framework for service allocation rather than a solver-specific optimization model. Consequently, the framework does not impose a single optimization strategy, but instead allows the integration of different optimization, heuristic, metaheuristic, machine learning, or multi-criteria decision-making approaches depending on the requirements, complexity, and constraints of specific IoT application scenarios. The selection and implementation of a concrete solver strategy therefore remain application-dependent and represent an important direction for future experimental research and implementation.

6. Scenario-based evaluation of the proposed framework

To further demonstrate the applicability of the proposed QoS-aware decision-making framework, a smart tram monitoring use-case scenario was developed within the context of a smart city tram transportation system (Figure 7). The objective of this evaluation is not to model all possible deployment configurations and communication conditions, but to illustrate how the proposed framework supports service allocation decisions across CRU, Fog/Edge, and Cloud layers under different processing and communication strategies. The evaluation focuses on the impact of service allocation on key QoS indicators, including service execution time, energy consumption, throughput, and network coverage.

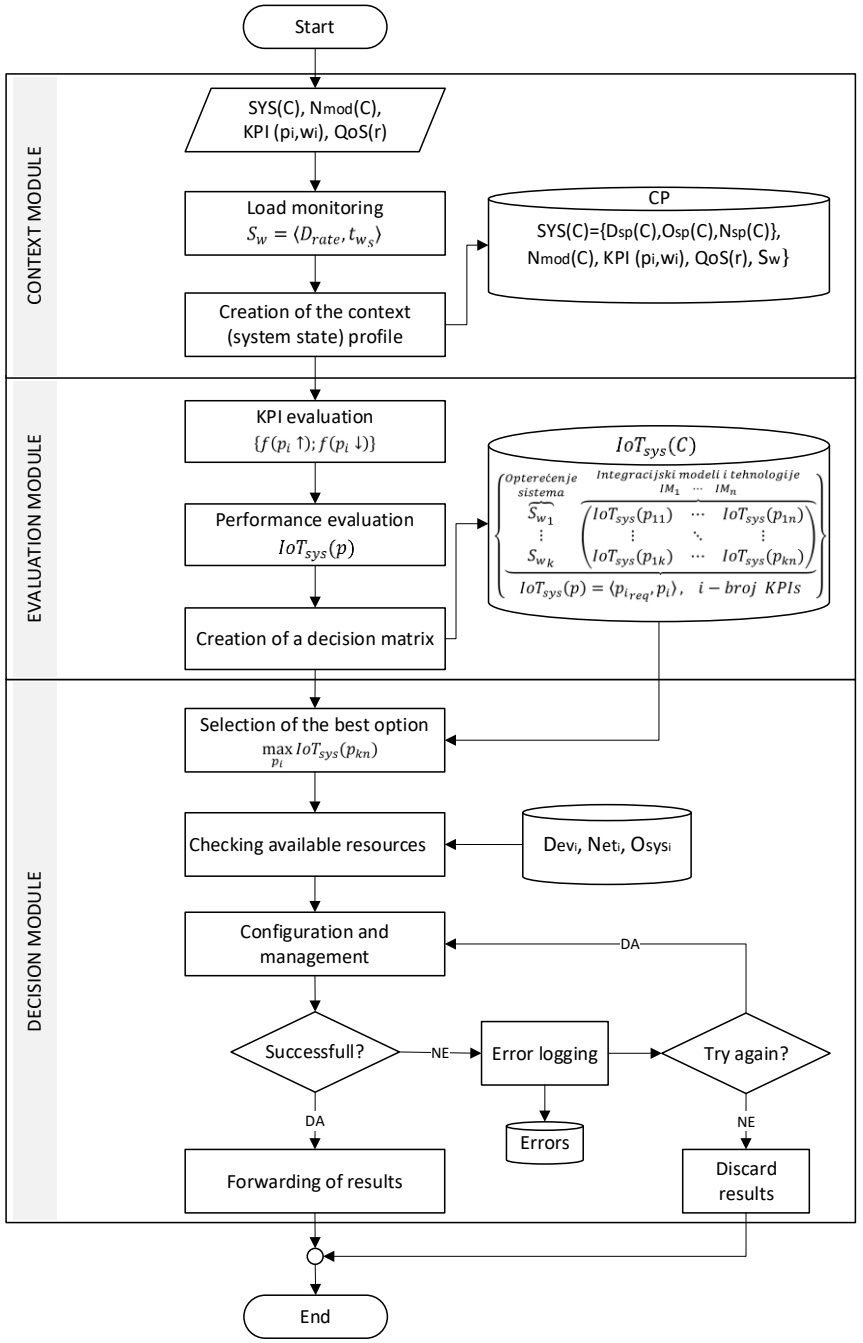

Figure 6. Algorithm for decision-making on service allocation

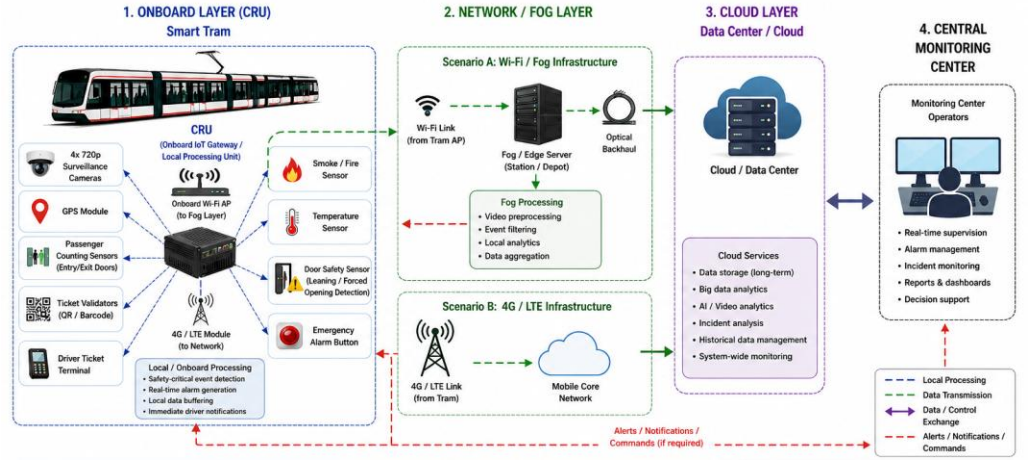

Figure 7. IoT architecture and communication scenarios for smart tram monitoring systems

The considered application represents a smart tram transportation system equipped with multiple IoT devices and monitoring components, including four 720p surveillance cameras, a GPS positioning module, passenger counting sensors, ticket validation devices, a driver terminal, smoke and fire detection sensors, temperature sensors, door safety sensors, and an emergency alarm button. These devices continuously generate heterogeneous operational, monitoring, and safety-related data that may be processed locally at the CRU or offloaded to Fog/Edge or Cloud infrastructure depending on the selected service allocation strategy.

For evaluation purposes, the generated traffic is classified into two categories: safety-critical sensor data and video surveillance data. Safety-critical services, including smoke/fire detection, temperature monitoring, door safety monitoring, and emergency alarm activation, are latency-sensitive and may be processed at the CRU, Fog, or Cloud layer. In contrast, video surveillance services generate substantially larger communication loads and require more computational resources, making them suitable for Fog/Edge or Cloud-based processing. Although passenger counting, GPS tracking, and ticket validation services generate comparatively lower traffic volumes, they remain important for operational supervision and transportation-system monitoring.

All devices initially transmit data to a Central Routing and Processing Unit (CRU) installed within the tram. The CRU may process part of the collected data locally or forward the data to external Fog/Edge or Cloud infrastructure through the available communication network. Consequently, the CRU represents the local IoT processing layer within the proposed architecture.

6.1. Use-case scenario and evaluation setup

For evaluation purposes, the generated traffic is classified into two categories: safety-critical sensor data and video surveillance data. Safety-critical sensor data represent approximately 10% of the total generated traffic and include services such as smoke/fire detection, temperature monitoring, door safety monitoring, emergency alarms, GPS tracking, passenger counting, and ticket validation events. Video surveillance data represent the remaining 90% of the generated traffic and originate from continuous onboard camera streams. Safety-critical sensor data may be processed at the CRU, Fog, or Cloud layer depending on the selected scenario, while video surveillance data are assumed to require computationally intensive processing and are therefore processed only at the Fog or Cloud layer. The processing results of safety-critical sensor data must be delivered to the CRU to support immediate driver notification and emergency response, whereas video analytics results are delivered to the monitoring center supported by the Cloud/data center infrastructure.

Different communication technologies and service allocation approaches were considered in order to evaluate the flexibility and effectiveness of the proposed framework. The following service allocation scenarios were analyzed:

1. **S1 (CLD-WF):** All safety-critical sensor data and video surveillance data are transmitted through the onboard Wi-Fi gateway and station AP/Fog relay infrastructure toward the Cloud/data center, where all processing and analytics are performed. The obtained safety-critical notifications are returned to the CRU.

2. **S2 (FOG-WF):** All safety-critical sensor data and video surveillance data are transmitted through the onboard Wi-Fi gateway to the station-based Fog node, where data processing is performed. The obtained safety-critical notifications are returned to the CRU, while video analytics results are forwarded to the monitoring center for storage and supervision.
3. **S3 (CRU-CLD-WF):** Safety-critical sensor data are processed locally on the CRU, while video surveillance data are transmitted through the onboard Wi-Fi gateway and AP/Fog relay infrastructure toward the Cloud/data center, where video processing and analytics are performed.
4. **S4 (CRU-FOG-WF):** Safety-critical sensor data are processed locally on the CRU, while video surveillance data are transmitted through the onboard Wi-Fi gateway toward the Fog/AP infrastructure for video pre-processing and filtering before being forwarded to the Cloud/data center for storage and advanced analytics.
5. **S5 (CRU-CLD-LTE):** Safety-critical sensor data are processed locally on the CRU, while video surveillance data are transmitted through the onboard LTE gateway toward the monitoring center, where video processing and storage are performed within the Cloud/data center infrastructure.
6. **S6 (CLD-LTE):** All safety-critical sensor data and video surveillance data are transmitted through the onboard LTE gateway toward the monitoring center, where all processing and analytics are performed within the Cloud/data center environment. The obtained safety-critical notifications are returned to the CRU.
7. **S7 (FOG-CLD-WF):** All safety-critical sensor data and video surveillance data are transmitted through the onboard Wi-Fi gateway to the station-based Fog node, where safety-critical sensor data are processed to generate low-latency alerts and operational notifications returned to the CRU. Simultaneously, video surveillance data are forwarded through the optical backhaul toward the Cloud/data center infrastructure, where video processing, analytics, and long-term storage are performed.

In the considered urban transportation environment, Wi-Fi-based communication was modeled as a shared access segment in which multiple trams located within the coverage area of the same station AP/Fog infrastructure simultaneously share the available communication capacity. Consequently, the effective transmission time in Wi-Fi scenarios depends on the aggregate traffic generated by all active trams within the considered communication domain. In contrast, LTE-based communication scenarios were modeled assuming realistic urban cellular deployment conditions in which trams may operate within the coverage areas of different LTE cells or sectors during movement through the transportation network. Therefore, LTE communication was modeled using representative per-tram uplink and downlink throughput values without strict aggregation of the generated traffic within a single shared LTE cell.

For the purposes of the evaluation, each tram is assumed to generate approximately 8 MB of input data during the considered transmission interval. The communication model assumes a packet size of 1024 bytes, while the total packet size including IP and UDP headers is 1052 bytes. The communication data rate is assumed to be 8000 kbps, resulting in approximately 7813 packets per transmission interval. In addition, a response payload of 10 kB is considered to represent alarm notifications, summarized operational information, or safety-critical feedback returned to the CRU after Fog or Cloud processing.

Based on experimental throughput measurements reported for IEEE 802.11n wireless networks, a representative throughput value of 50 Mbps was adopted for Wi-Fi-based communication scenarios (Yeoh, 2010). For LTE communication, average downlink and uplink throughput values of approximately 35 Mbps and 20 Mbps, respectively, were obtained through field measurements in the considered urban environment and subsequently adopted as representative LTE communication parameters. Fog/Edge nodes are interconnected with the monitoring center through high-speed optical infrastructure with an assumed throughput of 1 Gbps. The selected communication parameters were chosen to reflect realistic operating conditions in urban public transportation systems.

For the analytical estimation of computation time, the CRU was modeled as a lightweight onboard processing platform with a 1.4 GHz single-core processor. The Fog/Edge layer was represented by a medium-capacity edge node equipped with a 2.1 GHz quad-core processor, while the Cloud layer was modeled as a higher-capacity server with an 8-core processor operating at 3.6 GHz. Although processor frequency and the number of CPU cores do not fully capture the complete performance of real systems, these representative configurations provide a consistent basis for comparative evaluation of processing capabilities and service execution times across different service allocation strategies.

6.2. Selection and weighting of QoS key performance indicators

The key performance indicators (KPIs) considered in this evaluation were selected to capture the most important performance, communication, and deployment characteristics of the proposed layered IoT architecture. The selected KPIs include service execution time, energy consumption, network throughput, and network coverage. Together, these indicators provide a representative basis for evaluating the effectiveness of different service allocation strategies across the CRU, Fog/Edge, and Cloud layers.

Service execution time represents the overall end-to-end time required for successful service completion. It includes both network transmission time and computation time associated with the selected processing infrastructure. This KPI is particularly important in smart transportation systems due to the need for real-time monitoring, rapid incident detection, emergency response, and timely delivery of operational information.

Energy consumption reflects the computational and communication-related energy requirements of the onboard CRU infrastructure. Although the tram provides a permanent power supply, practical operation has shown that voltage instability, temporary power fluctuations, and emergency operating conditions may lead to instability, degraded performance, or unexpected resets of onboard processing and communication devices. Consequently, reducing computational and communication-related energy consumption contributes not only to lower operational power requirements, but also to improved stability, reliability, and continuous operation of the onboard IoT infrastructure. In the presented evaluation, the energy consumption of the CRU is analyzed with respect to both local data processing and wireless data transmission activities performed through Wi-Fi and LTE communication technologies under different service allocation scenarios.

Network throughput represents the effective communication capacity available for data transmission between the CRU, Fog/Edge infrastructure, and the Cloud-supported monitoring center. This KPI is particularly relevant in scenarios involving continuous video surveillance streams and high-volume sensor traffic, where insufficient throughput may result in communication bottlenecks, increased delays, and reduced scalability. It is represented using the effective uplink throughput of the communication technology used in each scenario. This choice is justified by the fact that the dominant traffic flow in the considered smart tram monitoring system is generated onboard and transmitted from the CRU toward the Fog/AP or Cloud-supported monitoring infrastructure. Downlink throughput was not selected as a primary KPI because the returned safety-critical notifications are comparatively small in size and do not represent the dominant communication load.

Network coverage reflects the availability and continuity of communication connectivity during tram operation. Since the considered architecture relies on Wi-Fi or LTE communication technologies, coverage directly influences service availability, monitoring continuity, and the reliability of safety-critical data transmission.

To determine the relative importance of the selected QoS criteria, the Fuzzy Analytic Hierarchy Process (FAHP) was employed. The method provides a structured mechanism for evaluating the relative importance of QoS criteria and deriving consistent weighting factors for subsequent multi-criteria analysis. FAHP was selected because it enables systematic weighting of multiple decision criteria while accounting for uncertainty and imprecision commonly associated with expert judgments. The pairwise comparison matrix was constructed based on expert assessments performed by the authors, drawing on their experience with real-world IoT-based transportation monitoring systems whose operational characteristics closely resemble the considered smart tram scenario. The evaluation primarily considered the expected impact of each KPI on service quality, operational reliability, and communication performance within the proposed layered IoT architecture.

The relative importance of the selected QoS key performance indicators, determined using the FAHP method, is presented in Table 1. The obtained FAHP weights indicate that service execution latency represents the most influential decision criterion with a weight, followed by network coverage, network throughput, and energy consumption.

KPI	Weight	Rank
Service execution latency	0.537	1
Network coverage	0.309	2
Network throughput	0.123	3
Energy consumption	0.031	4

Table 1. FAHP-derived weighting factors for the selected QoS key performance indicators

These results suggest that timely service delivery and communication availability are considered significantly more important than energy-related aspects in the considered transportation monitoring environment, reflecting the priority of maintaining reliable operation of safety-critical and real-time services. These weighting factors represent the relative importance of the considered QoS criteria and provide the basis for the subsequent utility-based multi-criteria evaluation of the analyzed service allocation scenarios. The analytical models, assumptions, and evaluation results associated with each KPI are presented in the following subsections.

6.2.1. Service execution time analysis

Service execution time was estimated as the sum of network transmission time and computation time required to obtain the corresponding processing results at their functional destinations. For safety-critical sensor data, the result must be available at the CRU to support immediate driver notification and emergency response, while video analytics results must be available at the monitoring center supported by the Cloud/data center infrastructure. In scenarios where Fog processing produces outputs for both destinations and these outputs are transmitted in parallel, the overall service execution time is determined by the slower communication path:

$$T_{service} = \max\left(T_{CRU\_result}, T_{monitoring\_result}\right)$$

The analytical estimation of computation time was based on the widely adopted Mobile Edge Computing (MEC) processing model, where execution time is calculated as the ratio between the required computational workload and the available processing capacity of the selected infrastructure (Mao et al., 2017). Accordingly, the computation time was estimated as:

$$T_{comp} = \frac{D \times C}{f_{CPU} \times N_{cores}}$$

where Drepresents the amount of processed data (bit), Cdenotes the required number of CPU cycles per bit, $f_{CPU}$is the processor frequency, and N_{cores} represents the number of available CPU cores.

More specifically, the computation time was estimated according to the relationship between the generated data volume, the required number of CPU cycles per bit, and the available processor capacity expressed through CPU frequency and the number of processing cores. For the purposes of this evaluation, a representative computational workload of 1000 CPU cycles/bit was adopted, following commonly used assumptions in MEC and computation offloading studies (Čolaković, 2023; C. Wang et al., 2017). This value provides a simplified and consistent basis for comparative analysis of different service allocation scenarios. However, more precise estimation would require detailed profiling of different application types and data categories, particularly safety-critical sensor data and computationally intensive video surveillance data.

Network transmission time was estimated using an analytical communication model based on the ratio between transmitted data volume and available network bandwidth, similarly to previous IoT offloading and performance evaluation studies (Čolaković, 2023). In addition to the transmission delay component, a fixed urban optical WAN/backhaul latency of 30 ms was considered in scenarios involving communication between Fog/AP infrastructure and the central monitoring/data center infrastructure. This value was introduced to approximate propagation, routing, and backbone communication delays in the considered urban environment.

For network transmission estimation, the generated communication traffic was modeled at the packet level. An application packet size of 1024 bytes was assumed, while the total transmitted packet size including IP and UDP headers was 1052 bytes. Based on the generated input data volume of 8 MB per tram, the estimated maximum number of transmitted packets was approximately 7813 within the observed transmission interval. The transmission delay was then calculated using the effective throughput of the corresponding communication technology and the communication load generated by simultaneously active trams within the considered communication segment. This packet-level approach provides a more realistic approximation than a simplified continuous bitrate model, particularly for wireless IoT scenarios involving packet overhead, shared medium access, and multiple active communication nodes.

Figure 8 illustrates the estimated service execution times for the analyzed service allocation scenarios as a function of the number of simultaneously active trams, highlighting the impact of processing location and communication infrastructure on overall service responsiveness.

Figure 8. Service execution time comparison for different service allocation scenarios and numbers of simultaneously active trams

The obtained results demonstrate that service execution time is strongly affected by both the selected processing layer and the communication technology. Scenarios that require intensive Fog-based processing of the complete generated traffic, particularly S2 (FOG-WF) and S4 (CRU-FOG-WF), produced the highest execution times. For six simultaneously active trams, the execution time reached 6739.3 ms in S2 and 6068.4 ms in S4, indicating that the Fog layer becomes a bottleneck when it is responsible for processing large volumes of video surveillance data in addition to safety-critical sensor traffic.

The best performance under low-load conditions was achieved by S7 (FOG-CLD-WF), with 475.0 ms for one tram. This scenario processes safety-critical sensor data at the Fog layer, while video surveillance data are forwarded to the Cloud/data center infrastructure for processing and storage. This confirms the advantage of separating latency-sensitive services from computationally intensive video analytics. However, as the number of simultaneously active trams increases, the execution time in S7 also increases due to shared Wi-Fi communication and the forwarding of video traffic toward the Cloud, reaching 2525.0 ms for six trams.

Scenario S3 (CRU-CLD-WF) also shows favorable performance because safety-critical sensor data are processed locally at the CRU, while only video surveillance data are transmitted toward the Cloud/data center infrastructure. Its execution time increases from 776.9 ms for one tram to 2425.5 ms for six trams, indicating better scalability than full Fog-processing scenarios. This result confirms that local processing of low-volume, latency-sensitive data can reduce the dependency on remote processing and improve responsiveness.

LTE-based scenarios show different behavior. S5 (CRU-CLD-LTE) and S6 (CLD-LTE) have higher initial communication delay due to lower LTE uplink throughput and additional cellular/core-network latency. However, their degradation with increasing numbers of trams is less pronounced because LTE communication was modeled using representative per-tram throughput values under urban multi-cell deployment assumptions. As a result, S5 becomes one of the most scalable scenarios under higher traffic loads, reaching 1873.1 ms for six trams.

Overall, the results confirm that no single service allocation scenario is optimal under all conditions. The ranking of scenarios changes depending on the number of simultaneously active trams, communication technology, and processing distribution. This supports the need for a QoS-aware decision-making framework capable of selecting the most suitable service allocation strategy according to current operational conditions and KPI priorities.

6.2.2. Energy consumption analysis

The estimation of CRU energy consumption was based on the analytical energy-consumption model proposed in (Čolaković, 2023), which considers both local data processing and wireless data transmission activities. For local data processing, the power-consumption parameters reported in (Bozorgchenani et al., 2019) were

adopted, where the CRU requires ($P_L = 900$ mW) during active processing and ($P_{idle} = 10$ m) during idle operation. The energy associated with wireless data transmission was estimated separately for Wi-Fi and LTE communication technologies. For Wi-Fi communication, the transmission-related power model parameters were assumed as $p_u = 283.17\ mW/Mbps$ and $\beta = 132.86$ mW, while for LTE communication the adopted parameters were $p_u = 438.39$ mW/Mbps and $\beta = 1288.04$ W (Huang et al., 2012). These parameters were used to comparatively evaluate the influence of different service allocation strategies and communication technologies on the operational energy requirements of the CRU.

Figure 9 presents the estimated CRU energy consumption obtained for the analyzed service allocation scenarios under different traffic loads represented by the number of simultaneously active trams. The obtained energy-consumption results demonstrate a strong dependence on the selected processing location and communication technology. The highest energy consumption was observed in scenarios S3 (CRU-CLD-WF) and S4 (CRU-FOG-WF), where safety-critical sensor data are processed locally at the CRU. In these scenarios, local computation dominates the overall energy consumption, resulting in values exceeding 516 J, which is more than 40 times higher than the energy consumption observed in the remaining scenarios. For this reason, S3 and S4 were excluded from the graphical presentation (Figure X) in order to preserve readability and improve visual interpretation of the comparative results.

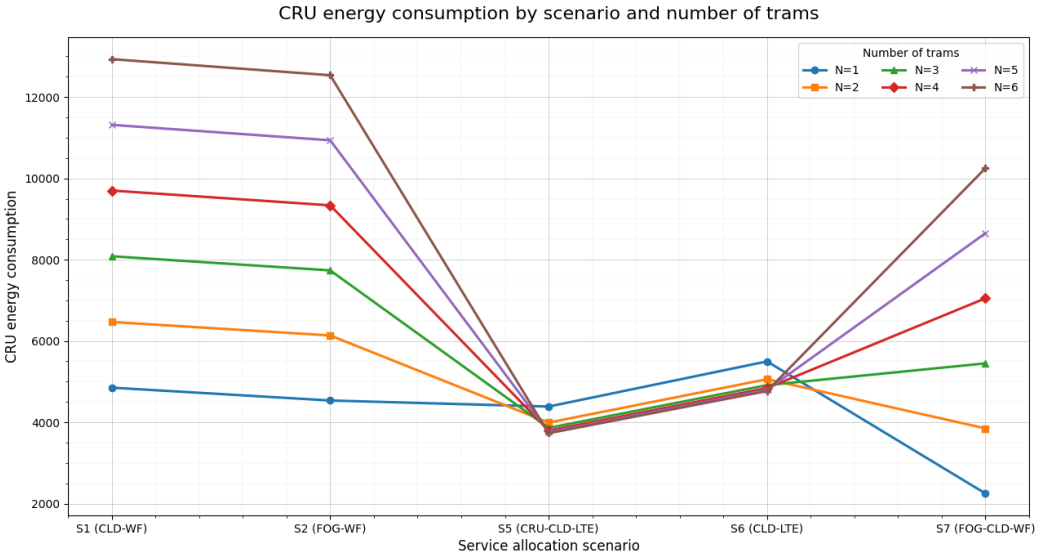

Figure 9. CRU energy consumption comparison for selected service allocation scenarios and numbers of simultaneously active trams

The lowest energy consumption was achieved by S7 (FOG-CLD-WF) under low traffic conditions, with a value of 2250 mJ for a single active tram. This scenario minimizes computational workload at the CRU by offloading all generated data to the Fog layer, where safety-critical sensor data are processed and video surveillance data are forwarded toward the Cloud/data center infrastructure. Similarly, S1 (CLD-WF) and S2 (FOG-WF) also demonstrate relatively low energy requirements because they eliminate local CRU processing and primarily rely on Wi-Fi-based transmission.

The LTE-based scenarios exhibit different behavior. Due to the higher transmission-related power coefficients associated with LTE communication, S6 (CLD-LTE) initially produces the highest transmission-related energy consumption among the offloading-oriented scenarios, reaching 5497.9 mJ for a single active tram. However, under the adopted communication model, the effective LTE transmission load assigned to an individual tram decreases as the number of simultaneously active trams increases, resulting in a gradual reduction of the estimated CRU transmission energy. Consequently, the energy consumption of S5 (CRU-CLD-LTE) decreases from 4387.5 mJ to 3731.3 mJ, while S6 (CLD-LTE) decreases from 5497.9 mJ to 4768.7 mJ as the number of active trams increases from one to six.

Overall, the obtained results indicate that local processing represents the dominant contributor to CRU energy consumption within the considered analytical model. The results further demonstrate that service allocation strategies that reduce computational workload at the CRU can significantly improve energy

efficiency. Therefore, from an energy-consumption perspective, scenarios based on Fog- and Cloud-assisted processing are generally more favorable than scenarios relying on local CRU processing of safety-critical data.

6.2.3. Network throughput analysis

Network throughput was considered as a communication-capacity KPI and represented using the effective uplink throughput of the communication technology employed in each service allocation scenario. This choice is justified by the fact that the dominant traffic flow in the considered smart tram monitoring system originates at the CRU and is transmitted toward Fog/AP or Cloud-supported monitoring infrastructure. Since the returned safety-critical notifications are comparatively small, downlink throughput was not considered a primary evaluation parameter.

For Wi-Fi-based scenarios, throughput was estimated as an effective per-tram throughput because the station-based AP/Fog infrastructure represents a shared communication medium. Consequently, the available Wi-Fi throughput was proportionally divided among simultaneously active trams. In contrast, LTE-based scenarios were evaluated using representative per-tram uplink throughput values, assuming an urban multi-cell deployment in which different trams may be served by different LTE cells or sectors.

Figure 10 presents the effective uplink throughput obtained for the analyzed service allocation scenarios and different numbers of simultaneously active trams.

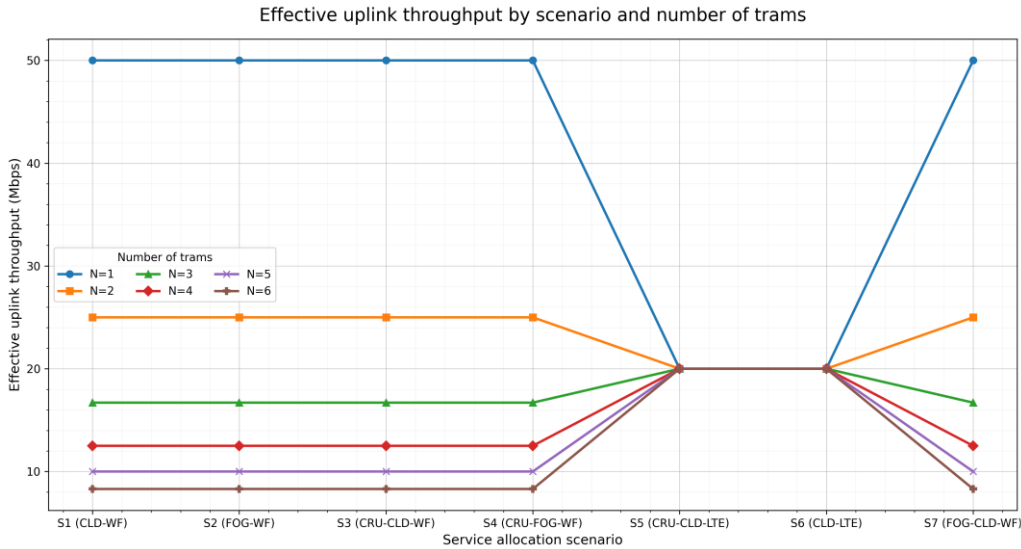

Figure 10. Effective uplink throughput comparison for different service allocation scenarios and numbers of simultaneously active trams

The results show that throughput is primarily determined by the characteristics of the communication infrastructure rather than by the selected processing layer. Under low-load conditions, all Wi-Fi-based scenarios achieved an effective throughput of 50 Mbps, significantly outperforming the LTE-based scenarios, which provide a representative uplink throughput of 20 Mbps. This higher communication capacity directly supports the transmission of high-volume video surveillance traffic and large-scale IoT data streams generated within the tram environment. However, because the Wi-Fi infrastructure represents a shared communication medium, the effective throughput available to each tram decreases as the number of simultaneously active trams increases. Consequently, the effective Wi-Fi throughput decreases from 50 Mbps for a single tram to 8.3 Mbps when six trams simultaneously share the same AP/Fog communication segment. In contrast, the LTE-based scenarios maintain a constant representative throughput of 20 Mbps under the adopted urban multi-cell deployment assumption.

These results indicate that Wi-Fi communication provides superior throughput performance under low and moderate traffic conditions, while LTE communication becomes more advantageous as network utilization increases. More specifically, Wi-Fi-based scenarios outperform LTE when one or two trams are simultaneously active, whereas LTE provides higher effective throughput when three or more trams operate

within the same Wi-Fi coverage area. Consequently, throughput represents not only a communication-performance indicator, but also an important measure of scalability and communication-resource availability within the proposed layered IoT architecture.

6.2.4. Network coverage analysis

Network coverage was considered as a deployment-level KPI reflecting the availability and continuity of communication connectivity along the tram transportation corridor. LTE-based scenarios were assumed to provide continuous communication coverage throughout the considered urban operating area due to the existing cellular infrastructure and mobility-oriented communication support. In contrast, Wi-Fi/Fog-based scenarios were assumed to provide approximately 80% route coverage because of the station-based deployment of AP/Fog infrastructure and the limited practical outdoor communication range of IEEE 802.11n technology. In the considered real-world tram transportation environment, AP/Fog nodes are assumed to be installed at tram stations, where the distance between neighboring stations typically ranges from approximately 500 m to 800 m. As a result, continuous communication coverage cannot be guaranteed along the entire route, and intermittent communication gaps may occur between adjacent station coverage areas, particularly along longer route segments. Consequently, Wi-Fi-based communication provides only partial corridor coverage compared to LTE-based communication, which is designed to support continuous connectivity during vehicle mobility.

Although Wi-Fi-based scenarios generally provide higher throughput under low and moderate traffic conditions, their coverage is inherently limited by the deployment density and geographical distribution of station-based AP/Fog infrastructure. As a result, temporary communication interruptions may occur when trams move outside the coverage areas of individual access points.

In contrast, LTE-based scenarios provide continuous connectivity across the entire considered transportation corridor. This characteristic is particularly important for safety-critical monitoring services, emergency notifications, and real-time operational supervision, where communication availability may be more important than achieving the highest possible throughput. Continuous network coverage also improves service reliability by reducing the probability of communication interruptions during tram movement.

Therefore, network coverage represents an important deployment-related QoS indicator that complements the performance-oriented KPIs considered in this study. While Wi-Fi/Fog-based communication can offer superior throughput and lower transmission costs in certain operating conditions, LTE-based communication provides greater connectivity continuity and operational reliability. Consequently, the selection of an appropriate service allocation strategy should consider not only latency, energy consumption, and throughput, but also the availability of communication infrastructure along the transportation route.

6.3. Utility-based evaluation and discussion

While the individual KPI analyses provide valuable insight into specific performance aspects of the considered service allocation scenarios, practical deployment decisions typically require the simultaneous consideration of multiple and often conflicting objectives. For example, a scenario that minimizes service execution time may not necessarily achieve the lowest energy consumption, while a communication technology offering higher throughput may provide lower network coverage. Therefore, selecting the most appropriate service allocation strategy requires a comprehensive multi-criteria evaluation capable of balancing different QoS requirements and operational priorities.

To address this challenge, the proposed QoS-aware framework incorporates utility-based decision-making using the normalized KPI values obtained from the analytical evaluation. The normalization process is performed through sigmoid utility functions, enabling heterogeneous performance indicators with different units and value ranges to be transformed into a common utility scale. The resulting utility values are subsequently combined using KPI weighting factors in order to calculate an overall utility score and rank the analyzed service allocation scenarios.

To evaluate the influence of KPI weighting strategies on the final service allocation decision, two weighting configurations were considered. The first configuration assigns equal importance to all KPIs (0.25 each), representing a balanced evaluation scenario without predefined priorities. The second configuration employs the weighting factors obtained through the FAHP procedure, reflecting the relative importance of the considered QoS criteria derived from expert assessment. The normalized KPI values were subsequently incorporated into the proposed utility-based decision-making model in order to rank the analyzed service allocation scenarios and assess the adaptability of the proposed framework under different decision-making priorities.

The normalized KPI values were incorporated into the proposed utility-based decision-making model in order to calculate the overall utility score for each service allocation scenario. Figure 11 presents the obtained utility scores for different numbers of simultaneously active trams using the considered KPI weighting configuration. The results provide a comprehensive assessment of the analyzed scenarios by simultaneously considering service execution time, energy consumption, network throughput, and network coverage.

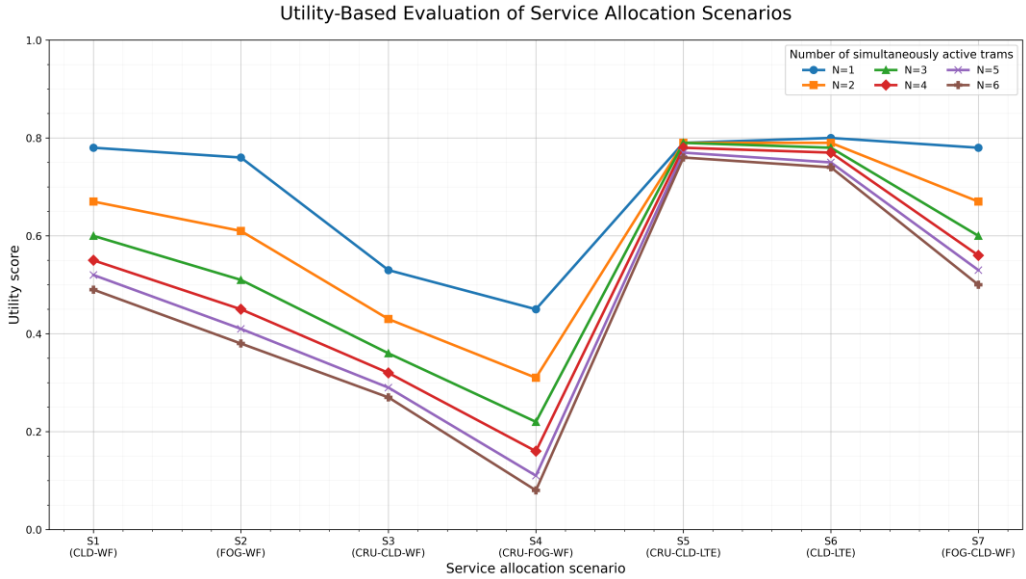

Figure 11. Utility-based scenario ranking using equal KPI weights

Figure 12 presents the utility-based ranking obtained using FAHP-derived KPI weights: service execution latency = 0.537, energy consumption = 0.031, network throughput = 0.123, and network coverage = 0.309.

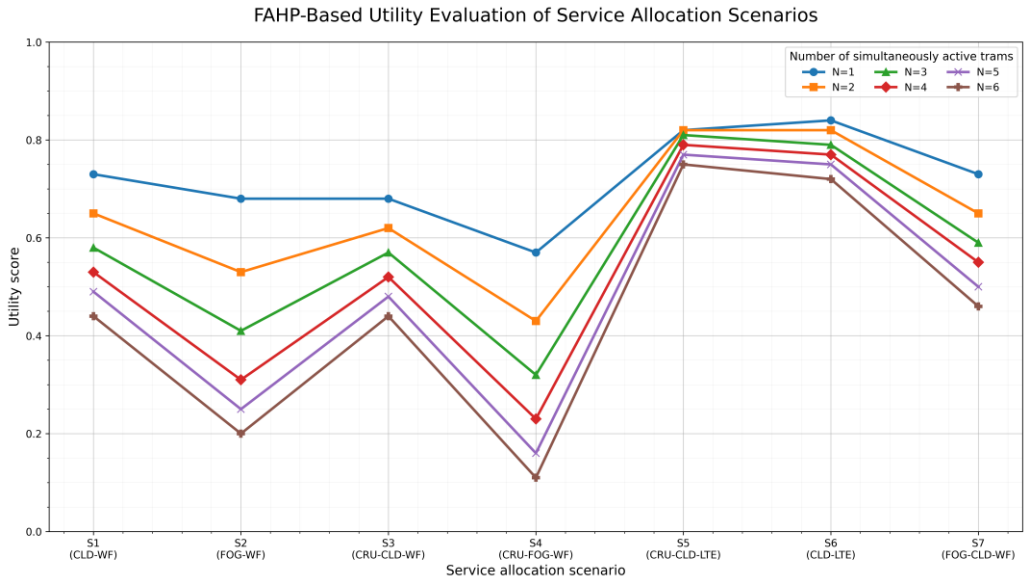

Figure 12. Utility-based scenario ranking using FAHP-derived KPI weights

The results obtained using equal KPI weights (0.25 for each criterion) indicate that the LTE-based scenarios S5 and S6 consistently achieve the highest utility scores across all considered traffic loads. For a single active tram, S6 achieves the highest utility score (0.80), closely followed by S5 (0.79), while the hybrid Wi-Fi-based scenario S7 achieves a comparable score of 0.78. However, as the number of simultaneously active trams increases, the utility scores of Wi-Fi-based scenarios decrease more rapidly due to the reduction of available throughput and the limited network coverage associated with the station-based AP/Fog infrastructure. Consequently, for higher traffic loads, S5 and S6 become the dominant alternatives.

The equal-weight evaluation further demonstrates that scenarios S3 and S4 consistently achieve the lowest utility scores. This behavior is primarily caused by their significantly higher service execution times under increasing communication and processing loads. In contrast, S7 performs well under low-load conditions because it achieves the lowest service execution time, but its overall ranking gradually deteriorates as throughput and coverage limitations become more influential.

When the FAHP-derived KPI weights are applied, the overall ranking remains generally consistent; however, the differences between scenarios become more pronounced. Since service execution latency (0.537) and network coverage (0.309) received the highest importance weights, the evaluation increasingly favors scenarios capable of maintaining low response times and continuous communication availability. As a result, S5 and S6 further strengthen their leading positions across all considered traffic loads. For example, under six simultaneously active trams, S5 and S6 maintain utility scores of 0.75 and 0.72, respectively, whereas the utility scores of S2 and S4 decrease to 0.20 and 0.11.

The comparison between the two weighting configurations clearly demonstrates the impact of decision-maker priorities on the final service allocation decision. Under equal weights, the evaluation provides a balanced consideration of all QoS criteria and allows hybrid Wi-Fi/Fog-Cloud solutions such as S7 to remain competitive under low-load conditions. In contrast, the FAHP-based evaluation places greater emphasis on latency and communication availability, thereby increasing the preference for LTE-based deployment strategies. This observation highlights an important characteristic of the proposed framework: the optimal service allocation strategy is not fixed, but depends on both operational conditions and the relative importance assigned to individual QoS criteria.

An additional observation is that the utility-based rankings differ from the conclusions that would be obtained by considering individual KPIs independently. For example, S7 achieved the best service execution time under low-load conditions, while S1 and S2 demonstrated favorable energy-consumption characteristics and Wi-Fi-based scenarios achieved the highest throughput under low traffic loads. Nevertheless, none of these scenarios consistently achieved the highest overall utility score once all QoS indicators were evaluated simultaneously. This confirms the necessity of multi-criteria decision-making and validates the applicability of the proposed QoS-aware decision-making framework for service allocation within layered IoT architectures.

7. Architectural styles for implementing the proposed mechanism

The proposed QoS mechanism can be implemented in software across all layers of the architecture, with the functions and complexity of implementation depending on the performance capabilities of the infrastructure at each layer.

7.1. Resources and requirements of IoT architectural layers

Given the varying performance levels and specific characteristics of the infrastructure within each layer, it is necessary to consider certain requirements and resources that constrain the possible architectural styles for the implementation.

IoT devices (perception layer) performs basic tasks such as data collection, initial processing, and data filtering. They are generally limited in terms of processing and memory capacity; therefore, only lightweight modules of the QoS mechanism can be processed on them (e.g., implementation of local rules for static decision-making, real-time control, etc.). Consequently, only simple software agents or services providing rules (e.g., latency or energy thresholds) can be deployed directly on these devices. The implementation strategy corresponds to distributed variant and hierarchical approaches described bellow, where a rapid local response is required with minimal overhead. For development and deployment of modules on IoT devices, the utilization of C, C++, and Rust programming languages is firmly required, given the efficient utilization of limited cumputing resources, memory safety, and their direct interaction with the hardware. Alternatively, whenever an IoT device resources allow, the MicroPython provides enhanced development and runtime environment for the higher level logic (Plauska et al., 2023).

Edge/Fog devices (gateways, microservers, MEC nodes) host the core of the dynamic decision-making approach, including their utility functions and the corresponding algorithms for multi-criteria decision-making.

The Edge/Fog nodes run "QoS brokers" or "QoS Decision Engines", which analyze KPIs and determine whether a task should remain locally processed (i.e., if they can complete the required services) or sould be further delivered to the Cloud. The main challenges at this layer involve ensuring system reliability in environments with variable loads and selecting the most appropriate approach.

For agile development and deployment, the implementation requires containerization of the software solution using Docker or Podman with lightweight orchestration, such as k3s or k0s (Z. Wang et al., 2022). The contenerization also supports different architectural styles, regardless of their implementation approach, e.g., distributed or centralized. Given ehnanced memory and computing resources compared to the perception layer, the utilization of the implementation framework and programming environment consenquently differs. The suitable programming languages include Pythondue to its extensive libraries for data processing, and Go for its high performance and ease of parallelization. However, when development of the more complex system services is required, or when high-performance Edge/Fog devices are available, the Node.js, C++, and Java, enable rapid implementation of complex algorithms while efficiently utilizing available resources.

The Cloud layer handles complex and resource-intensive processes, such as big data analytics, long-term storage, and machine learning (ML) model training. Within the proposed mechanism, the Cloud layer is responsible for centralized analysis and optimization, including the evaluation of strategies, updating of QoS profiles, and the global resource management. The dynamic updating of weighted KPI values and the prediction of system states under different conditions also executed on this layer, due to their resource-intensive requirements. Simlarily to the the Edge/Fog layer, the deployment allows contenerization of the software solution with enhanced platform capabilities, though. Multiple containers require dedicated tools for their orchestration (e.g., *Kubernetes*), following service registries (e.g., *Docker Registry*), and centralized metadata management (e.g., *PostgreSQL*). Additionally, when containers are deployed on Edge devices, utilization of KubeEdge provides their centralized orchestration (Mutichiro et al., 2021).

In terms of the development environment and programming languages for the implementation, the choice is less restricted compared to previous layers. The requirements are mostly related to scalability and ability of processing of large data volumes. The commonly used options including Python (for ML and analytics), Java/Scala (for distributed systems and big data frameworks such as Spark or Flink), and Go (for microservices and orchestration) (Bajaj et al., 2022). These languages also support integration with orchestration tools such as Kubernetes and KubeEdge.

7.2. Architectural styles

The proposed mechanism allows implementation using of the following architectural styles: Distributed (IoT–Edge/Fog–Cloud), Centralized (single QoS controller), Sharding-based (multiple controllers with identical roles), and Hierarchical (local–regional–central levels).

Distributed implementation across layers (IoT – Edge/Fog – Cloud) where each layer assumes the portion of services from the proposed mechanism that best matches its resources and the IoT application requirements (Fernando et al., 2025). For example, IoT devices running FreeRTOS/Zephyr OS execute local rules; Edge/Fog nodes use lightweight containerization (Docker/k3s) for the QoS broker; in the Cloud layer, a system is launched for orchestrating containerized applications on Edge devices (KubeEdge), as well as MLFlow for model training. In this way, the functional decomposition is provided, i.e., the appropriate modules are deployed on different devices. For example, IoT devices perform lightweight functions with static decision-making at the local level, Edge/Fog makes the main decision, while the Cloud processes long-term analytics, automates deployment, scaling, and container management, as well as the model training.

The advantages of the style include resource optimization, reduced latency, and improved scalability. The drawback lies in the complexity of coordination and synchronization, as well as in deciding which layers of the IoT architecture should be assigned to particular functional modules of the mechanism. The most suitable scenarios for this implementation are dynamic and heterogeneous environments, where both speed and global optimization of resources and performance are important.

Centralized implementation on a single dedicated device ("QoS controller") involves deployment of all decision-making modules on a single controller, such as an industrial controller (Memon et al., 2025). IoT and Edge/Fog devices only send data and execute commands based on feedback from the central controller.

The advantages of the style include straigforward deployment and maintenance, as well as the better synchronization across the system. However, its drawbacks lie in the performance limitations of the central

controller, includeing presence of the single point of failure (if an error occurs on the controller, the entire system is affected), and additional latency in communication with remote nodes. To mitigate the risks associated with a single point of failure, a certain solution for redudancy should be employed. Therefore, this style is most suitable for small to medium-sized IoT systems and environments with stable network connections.

Multiple dedicated devices with identical roles (horizontal sharding) involve deployment of several identical controllers, each responsible for its own shard (e.g., zone, service type, etc.) (Xiong et al., 2025). The main advantage of this style is improved scalability and resilience, without overload on a single controller. However, its drawbacks include challenges related to load balancing and service migration between shards.

The most suitable implementation of this style include smart city scenarios and large-scale IoT systems where each controller is responsible for a specific zone or one or more services. The example include intelligent transportation systems with thousands of sensors or urban IoT platforms.

Hierarchical style (local + regional + central controller) involves a multi-level control structure, where the local controller makes real-time decisions and manages IoT devices, following regional controllers oversee the Edge/Fog layer, and the central controller performs optimization and model training (Papaioannou et al., 2025). The central controller additionally sends feedback to the local and regional controllers enabling dynamic decision-making.

The main advantage of this style is the balance between speed and global optimization, including resilience to network disruptions. However, its drawback lies in the complex coordination required among controllers and the requirement for clearly defined priority rules. The most suitable scenarios for the implementation of the the mechanism in this style include complex distributed systems, such as Industry 4.0, energy management, and smart factories. Table 2 presents a summary of the advantages, disadvantages, and application areas of architectural styles.

The proposed architectural implementation styles are fully aligned with the architecture and the proposed decision-making mechanism. The three-layer structure (IoT devices – Edge/Fog – Cloud) is extended with the possibility of centralized or distributed mechanism implementation, where each layer utilizes appropriate platforms, operating systems, protocols, and development frameworks: IoT devices focus on minimalist agents and energy-efficient protocols, the Edge/Fog layer enables multi-criteria decision-making and dynamic load balancing, while the Cloud layer handles complex analytics tasks, long-term storage, and ML model training.

Style	Advantages	Disadvantages	When to use
Distributed (IoT–Edge/Fog–Cloud)	Scalability; reduced latency; load distribution	More complex coordination and synchronization between layers. Requires orchestration (k3s, Kubernetes) and KPI synchronization across layers.	Heterogeneous environments; when both fast local responses and global optimization are needed
Centralized on a single device ("QoS controller")	Simple maintenance and auditing; less code on IoT/Edge devices	Higher latency; limited scalability. Single point of failure—mitigated by redundant controllers.	Small and medium-sized systems; pilot projects; stable network environments
Multiple controllers with identical roles (sharding)	Scalability; resilience; fault isolation	Uneven load; service migration requires additional mechanisms. Requires load balancing and migration mechanisms (Consul, Raft).	Large-scale systems with many devices and services; massive IoT (smart cities, transportation)
Hierarchical model (local–regional–central)	Low latency at local level; global optimization; robustness against network interruptions	More complex coordination; need for clear priority rules. Requires clear division of responsibilities and coordination protocols.	Distributed systems (Industry 4.0, smart factories, energy sector)

Table 2. Architectural styles for implementing the QoS-aware mechanism

This distribution of tasks and technologies ensures that the proposed mechanism is flexible and adaptable to various scenarios, confirming that the proposed framework is not merely a theoretical model, but also practically applicable in real IoT environments. The proposed architectural styles largely overlap with traditional architectural styles used in software system architecture and design. Most of styles are sufficiently

adaptable for use across different IoT domains. Regardless of which style is applied, the key requirement lies in the identification of the quality attributes it should support, such as programmability, maintainability, and security. However, the choice of style must also reflect the specific needs of particular industrial applications in which it is deployed.

Finally, an additional requirement for selection and implementation of a certain architectural style depends on the requirement for dynamic service allocation that optimizes the selected QoS parameters. In this particular case the emphasis is on the total execution time of IoT services and the total energy consumed by the IoT device during service execution.

8. Conclusion and future research

Optimal service and resource allocation within layered IoT architectures remains one of the key challenges in achieving the desired Quality of Service (QoS) under heterogeneous computational and communication conditions. This paper analyzed the fundamental principles of integrating IoT devices with Edge, Fog, and Cloud infrastructures and proposed a QoS-aware decision-making framework for intelligent service allocation across different architectural layers. The framework combines QoS profiling, KPI-based evaluation, utility functions, and multi-criteria decision-making principles to support both static and dynamic service allocation strategies.

The proposed framework formalizes the service allocation process through the definition of key performance indicators, QoS profiles, and adaptive decision-making mechanisms capable of balancing multiple and often conflicting objectives. Unlike approaches that rely on computationally intensive optimization algorithms, the proposed solution provides a lightweight and flexible decision-making mechanism that can be implemented in heterogeneous IoT environments while maintaining low computational overhead. Furthermore, the framework is not limited to a specific optimization technique and can be integrated with advanced decision-support approaches, including machine learning and AI-based mechanisms.

To demonstrate its applicability, the framework was evaluated through a smart transportation use case involving multiple service allocation scenarios across CRU, Fog, and Cloud infrastructures. The results showed that different allocation strategies exhibit distinct trade-offs between service execution latency, energy consumption, throughput, and network coverage. The utility-based evaluation further demonstrated that the preferred service allocation strategy depends not only on current operating conditions but also on the relative importance assigned to individual QoS criteria. These findings confirm the necessity of multi-criteria decision-making and validate the ability of the proposed framework to support adaptive and context-aware service allocation in layered IoT environments. The proposed framework therefore provides both a conceptual foundation and a practical methodology for QoS-aware service allocation in next-generation IoT systems. Its modular design allows adaptation to different application domains, communication technologies, and deployment architectures, making it suitable for a wide range of smart-city, transportation, industrial, and cyber-physical system applications.

Future research should focus on the identification of application-specific QoS profiles and KPI weighting factors, the implementation and validation of the framework in simulation and real-world environments with real-time KPI monitoring, the investigation of dynamic service migration between CRU, Fog/Edge, and Cloud layers, and the evaluation of the framework in emerging communication environments including 5G, network slicing, and future 6G-enabled IoT architectures.

Overall, this research establishes a foundation for QoS-aware decision-making in layered IoT systems and provides a flexible framework that can support the development of intelligent, scalable, reliable, and resource-efficient IoT services capable of adapting to evolving operational requirements and technological environments.

References

Abdulazeez, D. H., & Askar, S. K. (2023). Offloading Mechanisms Based on Reinforcement Learning and Deep Learning Algorithms in the Fog Computing Environment. *IEEE Access*, *11*, 12555–12586. https://doi.org/10.1109/ACCESS.2023.3241881

Al-Awami, S. H., Mahfud Al-Aty, M., & Al-Najar, M. F. (2023). Comparison of IoT Architectures Based on the Seven Essential Characteristics. *2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, 305–310. https://doi.org/10.1109/MI-STA57575.2023.10169289

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Communications Surveys & Tutorials*, *17*(4), 2347–2376.

Almudayni, Z., Soh, B., Samra, H., & Li, A. (2025). Energy Inefficiency in IoT Networks: Causes, Impact, and a Strategic Framework for Sustainable Optimisation. *Electronics*, *14*(1), 159. https://doi.org/10.3390/electronics14010159

Alonso, R. S., Sittón-Candanedo, I., García, Ó., Prieto, J., & Rodríguez-González, S. (2020). An intelligent Edge-IoT platform for monitoring livestock and crops in a dairy farming scenario. *Ad Hoc Networks*, *98, 102047*, 1–23. https://doi.org/10.1016/j.adhoc.2019.102047

Ammar, M., Russello, G., & Crispo, B. (2018). Internet of Things: A survey on the security of IoT frameworks. *Journal of Information Security and Applications*, *38*, 8–27. https://doi.org/10.1016/j.jisa.2017.11.002

Babovic, Z. B., Protic, J., & Milutinovic, V. (2016). Web Performance Evaluation for Internet of Things Applications. *IEEE Access*, *4*, 6974–6992.

Bacanin, N., Zivkovic, M., Bezdan, T., Venkatachalam, K., & Abouhawwash, M. (2022). Modified firefly algorithm for workflow scheduling in cloud-edge environment. *Neural Computing and Applications*, *34*(11), 9043–9068. https://doi.org/10.1007/s00521-022-06925-y

Bajaj, K., Sharma, B., & Singh, R. (2022). Implementation analysis of IoT-based offloading frameworks on cloud/edge computing for sensor generated big data. *Complex & Intelligent Systems*, *8*(5), 3641–3658. https://doi.org/10.1007/s40747-021-00434-6

Bozorgchenani, A., Tarchi, D., & Corazza, G. E. (2019). Centralized and Distributed Architectures for Energy and Delay Efficient Fog Network-Based Edge Computing Services. *IEEE Transactions on Green Communications and Networking*, *3*(1), 250–263. https://doi.org/10.1109/TGCN.2018.2885443

Bu, T., Huang, Z., Zhang, K., Wang, Y., Song, H., Zhou, J., Ren, Z., & Liu, S. (2023). Task scheduling in the internet of things: Challenges, solutions, and future trends. *Cluster Computing*. https://doi.org/10.1007/s10586-023-03991-2

Chiang, M., & Zhang, T. (2016). Fog and IoT: An Overview of Research Opportunities. *IEEE Internet of Things Journal*, *3*(6), 854–864.

Čolaković, A. (2023). IoT systems modeling and performance evaluation. *Computer Science Review*, *50*, 1–26. https://doi.org/10.1016/j.cosrev.2023.100598

Čolaković, A., Karahodža, B., & Hasković Džubur, A. (2025). Perspective Chapter: QoS-Aware IoT Framework for Performance Control and Resource Management. In *Quality Control—Artificial Intelligence, Big Data, and New Trends [Working Title]*. IntechOpen. https://doi.org/10.5772/intechopen.1010421

Fernando, N., Shrestha, S., Loke, S. W., & Lee, K. (2025). On Edge-Fog-Cloud Collaboration and Reaping Its Benefits: A Heterogeneous Multi-Tier Edge Computing Architecture. *Future Internet*, *17*(1), 22. https://doi.org/10.3390/fi17010022

Firouzi, F., Farahani, B., & Marinšek, A. (2022). The convergence and interplay of edge, fog, and cloud in the AI-driven Internet of Things (IoT). *Information Systems*, *107*, 101840. https://doi.org/10.1016/j.is.2021.101840

Fröhlich, P., Gelenbe, E., Fiołka, J., Chęciński, J., Nowak, M., & Filus, Z. (2021). Smart SDN Management of Fog Services to Optimize QoS and Energy. *Sensors*, *21*(9), 3105. https://doi.org/10.3390/s21093105

Gasmi, K., Dilek, S., Tosun, S., & Ozdemir, S. (2022). A survey on computation offloading and service placement in fog computing-based IoT. *The Journal of Supercomputing*, *78*(2), 1983–2014. https://doi.org/10.1007/s11227-021-03941-y

Gill, S. S., Garraghan, P., & Buyya, R. (2019). ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices. *Journal of Systems and Software*, *154*, 125–138. https://doi.org/10.1016/j.jss.2019.04.058

Haidri, R. A., Katti, C. P., & Saxena, P. C. (2020). Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. *Journal of King Saud University - Computer and Information Sciences*, *32*(6), 666–683. https://doi.org/10.1016/j.jksuci.2017.10.009

Hu, H., Song, W., Wang, Q., Hu, R. Q., & Zhu, H. (2022). Energy Efficiency and Delay Tradeoff in an MEC-Enabled Mobile IoT Network. *IEEE Internet of Things Journal*, *9*(17), 15942–15956. https://doi.org/10.1109/JIOT.2022.3153847

Huang, J., Qian, F., Gerber, A., Mao, Z. M., Sen, S., & Spatscheck, O. (2012). A close examination of performance and power characteristics of 4G LTE networks. *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys '12*, 225–238. https://doi.org/10.1145/2307636.2307658

IoT Application Modules Placement and Dynamic Task Processing in Edge-Cloud Computing. (2020). *IEEE Internet of Things Journal*, *8*(16), 12771–12781. https://doi.org/10.1109/JIOT.2020.3007751

Kaushik, A., & Hamed, S. A.-R. (2022). A Hybrid Latency- and Power-Aware Approach for Beyond Fifth-Generation Internet-of-Things Edge Systems. *IEEE Access*, *10*, 87974–87989. https://doi.org/10.1109/ACCESS.2022.3200035

Khattak, K. S., & Khan, Z. H. (2024). Evaluation and Challenges of IoT Simulators for Intelligent Transportation System Applications. *Science, Engineering and Technology*, *4*(1), 94–111. https://doi.org/10.54327/set2024/v4.i1.107

Khebbeb, K., Hameurlain, N., & Belala, F. (2020). A Maude-based rewriting approach to model and verify Cloud/Fog self-adaptation and orchestration. *Journal of Systems Architecture*, *110, 10182*. https://doi.org/10.1016/j.sysarc.2020.101821

Madsen, H., Burtschy, B., Albeanu, G., & Popentiu-Vladicescu, Fl. (2013). Reliability in the utility computing era: Towards reliable Fog computing. *20th International Conference on Systems*, 43–46.

Mao, Y., You, C., Zhang, J., Huang, K., & Letaief, K. B. (2017). A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials*, *19*(4), 2322–2358. https://doi.org/10.1109/COMST.2017.2745201

Maray, M., & Shuja, J. (2022). Computation Offloading in Mobile Cloud Computing and Mobile Edge Computing: Survey, Taxonomy, and Open Issues. *Mobile Information Systems*, *2022*, 1–17. https://doi.org/10.1155/2022/1121822

Memon, S. A., Andriukaitis, D., Navikas, D., Markevicius, V., Valinevicius, A., Zilys, M., Prauzek, M., Konecny, J., Brida, P., & Li, Z. (2025). Centralized and Distributed Controller Placement in Terms of Scalability and Fault Tolerance: A Review. *2025 29th International Conference on Methods and Models in Automation and Robotics (MMAR)*, 449–454. https://doi.org/10.1109/MMAR65820.2025.11150828

Mutichiro, B., Tran, M.-N., & Kim, Y.-H. (2021). QoS-Based Service-Time Scheduling in the IoT-Edge Cloud. *Sensors*, *21*(17), 5797. https://doi.org/10.3390/s21175797

Omoniwa, B., Hussain, R., Javed, M. A., Bouk, S. H., & Malik, S. A. (2019). Fog/edge computing-based IoT (FECIoT): Architecture, applications, and research issues. *IEEE Internet of Things Journal*, *6*(3), 4118–4149. https://doi.org/10.1109/JIOT.2018.2875544

Papaioannou, C., Dimara, A., Papaioannou, A., Tzitzios, I., Anagnostopoulos, C.-N., & Krinidis, S. (2025). Hierarchical Resources Management System for Internet of Things-Enabled Smart Cities. *Sensors*, *25*(3), 616. https://doi.org/10.3390/s25030616

Plauska, I., Liutkevičius, A., & Janavičiūtė, A. (2023). Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. *Electronics*, *12*(1), 143. https://doi.org/10.3390/electronics12010143

Reddy, K. H. K., Luhach, A. Kr., Pradhan, B., Dash, J. K., & Roy, D. S. (2020). A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. *Sustainable Cities and Society*, *63*, 102428. https://doi.org/10.1016/j.scs.2020.102428

Safavat, S., Sapavath, N. N., & Rawat, D. B. (2020). Recent advances in mobile edge computing and content caching. *Digital Communications and Networks*, *6*(2), 189–194. https://doi.org/10.1016/j.dcan.2019.08.004

Sharma, A., & Thangaraj, V. (2024). Intelligent service placement algorithm based on DDQN and prioritized experience replay in IoT-Fog computing environment. *Internet of Things*, *25*, 101112. https://doi.org/10.1016/j.iot.2024.101112

Sinha Roy, D., Behera, R. K., Reddy, K. H. K., & Buyya, R. (2019). A Context-Aware Fog Enabled Scheme for Real-Time Cross-Vertical IoT Applications. *IEEE Internet of Things Journal*, *6*(2), 2400–2412. https://doi.org/10.1109/JIOT.2018.2869323

Tuli, S., Mahmud, R., Tuli, S., & Buyya, R. (2019). FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing. *Journal of Systems and Software*, *154*, 22–36. https://doi.org/10.1016/j.jss.2019.04.050

Wang, C., Liang, C., Yu, F. R., Chen, Q., & Tang, L. (2017). Computation Offloading and Resource Allocation in Wireless Cellular Networks With Mobile Edge Computing. *IEEE Transactions on Wireless Communications*, *16*(8), 4924–4938. https://doi.org/10.1109/TWC.2017.2703901

Wang, S., Song, X., Xu, H., Song, T., Zhang, G., & Yang, Y. (2023). Joint offloading decision and resource allocation in vehicular edge computing networks. *Digital Communications and Networks*. https://doi.org/10.1016/j.dcan.2023.03.006

Wang, Z., Goudarzi, M., Aryal, J., & Buyya, R. (2022). *Container Orchestration in Edge and Fog Computing Environments for Real-Time IoT Applications* (arXiv:2203.05161). arXiv. https://doi.org/10.48550/arXiv.2203.05161

Wei, H., Luo, H., Sun, Y., & Obaidat, M. S. (2020). Cache-Aware Computation Offloading in IoT Systems. *IEEE Systems Journal*, *14*(1), 61–72. https://doi.org/10.1109/JSYST.2019.2903293

Xavier, T. C. S., Santos, I. L., Delicato, F. C., Pires, P. F., Alves, M. P., Calmon, T. S., Oliveira, A. C., & Amorim, C. L. (2020). Collaborative resource allocation for Cloud of Things systems. *Journal of Network and Computer Applications*, *159, 10259*, 1–18. https://doi.org/10.1016/j.jnca.2020.102592

Xiong, X., Li, M., Yu, F. R., Zhang, H., Wang, K., & Si, P. (2025). Cloud-Edge-End Collaborative Computing-Enabled Intelligent Sharding Blockchain for Industrial IoT Based on PPO Approach. *IEEE Transactions on Mobile Computing*, *24*(9), 8011–8024. https://doi.org/10.1109/TMC.2025.3554568

Yeoh, C. Y. (2010). Experimental Study of 802.11n Network. *Proceedings of the 12th International Conference on Advanced Communication Technology*, 1067–1072.

Zahoor, S., & Mir, R. N. (2021). Resource management in pervasive Internet of Things: A survey. *Journal of King Saud University - Computer and Information Sciences*, *33*(8), 921–935. https://doi.org/10.1016/j.jksuci.2018.08.014

Zhang, J., Ma, M., He, W., & Wang, P. (2020). On-demand deployment for IoT applications. *Journal of Systems Architecture*, *111, 10179*. https://doi.org/10.1016/j.sysarc.2020.101794

Zhou, S., Jadoon, W., & Khan, I. A. (2023). Computing Offloading Strategy in Mobile Edge Computing Environment: A Comparison between Adopted Frameworks, Challenges, and Future Directions. *Electronics*, *12*(11), 1–30. https://doi.org/10.3390/electronics12112452