UDC 004.75 https://doi.org/10.20535/2786-8729.6.2025/334607

# DECENTRALIZED TASK ALLOCATION METHOD IN HIERARCHICAL IoT SYSTEMS USING FUZZY LOGIC

**Oleksandr Rolik***
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
http://orcid.org/0000-0001-8829-4645

**Dmytro Nahaiko**
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
https://orcid.org/0009-0003-3611-3605

*Corresponding author: o.rolik@kpi.ua

The use of fog and edge computing extends the computational capabilities of IoT systems to the network edge, contributing to the minimization of delays during task execution. Osmotic computing complements distributed computing by providing seamless integration between computational environments through dynamic migration of micro-elements across different hierarchy tiers according to current load conditions and resource availability. However, within the concept of osmotic computing, a key challenge remains the effective management of task allocation under conditions of uncertainty, dynamism, and heterogeneity of the IoT environment. The aim of this study is to improve the efficiency of resource utilization and task allocation in hierarchical IoT systems based on osmotic computing under uncertain and dynamically changing environmental conditions. The object of the study is the process of task allocation in multi-tier IoT systems that include cloud, fog, and edge computing. The subject of the study is methods and models for task allocation and computing resource management in IoT systems using the osmotic computing paradigm.

The paper presents a three-tier hierarchical management model built on cloud, fog, and edge environments, which implements a centralized-decentralized management approach. Each tier is represented by a set of computing nodes and a management system that performs local task allocation, resource state monitoring, and micro-element management. The management system of the lower tier is subordinate to the higher-tier management system in the hierarchy. A method for decentralized task allocation in hierarchical IoT systems using fuzzy logic has been developed. The allocation method includes two decision-making stages using a fuzzy inference system: determining the direction of task allocation and selecting the optimal computing node for its execution. The determination of task allocation direction is carried out based on task characteristics, and the suitability rating of computing nodes is determined considering task execution latency, resource utilization efficiency, and load balancing. The task is assigned to the node with the maximum rating. The use of fuzzy logic ensures rational decision-making under conditions of uncertainty in real-time, which is characteristic of highly heterogeneous and dynamic IoT environments.

Experimental modeling and investigation of the method were carried out using the iFogSim simulation environment. The research results show that the percentage of locally executed tasks remains virtually unchanged with different numbers of tasks, indicating stability in decision-making. Increasing the intensity of task generation leads to an increase in task computation latency due to increased load on computing nodes, while task assignment latency and response latency remain unchanged. The method demonstrated adaptability in task allocation for different types of tasks.

**Keywords:** Internet of Things, IoT, Fog Computing, Edge Computing, Osmotic computing, Fuzzy Logic

## 1. Introduction

The active development and implementation of IoT technologies have led to a rapid growth in the number of devices connected to IoT. In 2024 alone, their number exceeded 18 billion, which is 13%

more than in 2023, while corporate spending on IoT infrastructure reached USD 298 billion [1]. This trend is accompanied by an increase in data volumes generated by IoT devices and an increasing computational load on IT infrastructure, which complicates maintaining a guaranteed level of Quality of Service (QoS). Additionally, the high heterogeneity and dynamism that characterize IoT environments [2] require an appropriate level of adaptability in IoT systems to real-time changes.

The use of fog and edge computing partially addresses these problems by extending the system's computational capabilities to the network edge, thereby bringing data collection, analysis, and processing closer to the data source. This approach helps minimize data transmission delays, reduces overall task execution time, and decreases load in the cloud environment [3].

However, the application of fog and edge technologies creates new challenges related to uneven load distribution, limited resources at the periphery, and the need to respond to dynamic connections, disconnections, and failures of computing nodes.

To address these challenges, [4] proposed the concept of osmotic computing, inspired by the chemical process of osmosis, which involves the autonomous and dynamic management of computational resources. Through continuous vertical load balancing between cloud, fog, and edge environments, osmotic computing ensures system adaptation to real-time changes, maintaining uniform load distribution among different computing environments [4, 5].

However, considering the concept of osmotic computing, a key challenge remains the effective management of task allocation to improve the productivity and efficiency of distributed multi-tier IoT systems. It is important to ensure service delivery with an appropriate QoS level under conditions of uncertainty, dynamism, and heterogeneity of the environment.

## 2. Literature review

Task allocation in distributed IoT systems is a well-researched problem. [6] conducted a comprehensive analysis of various computing paradigms, including fog, edge, and osmotic computing, which contribute to latency reduction and optimization of resource utilization. Additionally, [6] examines the main groups of approaches and their applications for effective task and resource management, including metaheuristic, machine learning, and hybrid approaches. Among these, for example, [7] developed an approach for scheduling latency-sensitive tasks in a heterogeneous Fog-Cloud environment using a Multi-Level Feedback Queue (MLFQ) to classify tasks based on the priorities of each level. In [8], an improved fireworks algorithm is proposed to optimize load distribution in a fog environment. For edge computing, [9] implemented deep reinforcement learning for dynamic workload scheduling. In [10], a bio-inspired load balancing algorithm was developed using osmotic pressure principles.

In [5], scientific approaches using osmotic computing are systematized, which implement the dynamic distribution of tasks between the Edge, Fog, and Cloud environments. Particular attention is paid to the problems of self-organization, load detection, and adaptive scaling. The methods used include heuristic algorithms, graph theory models, fuzzy logic, and artificial neural networks. The results show the advantages of the osmotic model in conditions of limited resources and unpredictable network topology.

The use of fuzzy logic warrants special attention, as it allows consideration of input data inaccuracy and system state uncertainty. Approaches using fuzzy inference systems [11–13] have relatively low computational complexity and can respond quickly to changing conditions, making them promising for distributed IoT systems that encounter unpredictable and dynamic environmental changes.

In [14], a comprehensive approach is proposed for solving task scheduling and load balancing problems in a heterogeneous Fog-Cloud environment. A Binary Linear-Weight JAYA (BLWJAYA) algorithm was developed for optimal task mapping to computing nodes. Fuzzy logic is used to determine target tiers for task allocation, considering resource heterogeneity and system requirements (network bandwidth, task size, resource utilization, and latency sensitivity).

In [15], a task prioritization mechanism is presented, where tasks are classified according to the level of sensitivity to delay and served at different tiers. Decisions are made using fuzzy logic and

heuristic utility functions, which allows for ensuring a balance between task priority and resource availability.

In [16], a Dynamic Task Allocation using Fuzzy Logic Enhanced approach (DFA-FLE) is proposed, which adapts to environmental changes, ensuring latency reduction and improved resource utilization efficiency through a two-tier fuzzy inference system.

Researchers also actively study task scheduling approaches that combine fuzzy logic methods and machine learning [17] or heuristic strategies [18]. In [19], the effectiveness of multi-criteria decision-making is demonstrated when considering parameters such as bandwidth, latency, energy consumption, and task execution cost. In [20], an approach for workflow scheduling and allocation in hybrid Fog-Cloud computing environments using multi-agent systems and fuzzy logic is proposed.

To summarize, all these studies emphasize the importance of adaptive task allocation methods in distributed multi-tier IoT systems for improving efficiency, reducing latency, and optimizing resource utilization. However, existing solutions do not combine osmotic computing, which enables adaptive inter-tier load balancing, with fuzzy logic apparatus to ensure effective system operation in real-time considering environmental uncertainty. Additionally, the works consider only upward task flow and do not account for the possibility of task generation at higher hierarchy tiers with subsequent task allocation to the edge tier.

## 3. The aim and objectives of the study

The aim of the study is to improve the efficiency of resource utilization and task distribution in hierarchical IoT systems based on osmotic computing using fuzzy logic. The research focuses on ensuring the rational selection of a computing environment for performing tasks in conditions of uncertainty and dynamic environmental changes.

The object of the study is the processes of task allocation in IoT systems with multi-tier environments that include cloud, fog, and edge computing.

The subject of the study is the methods and models for task allocation and computing resource management in IoT systems using the osmotic computing paradigm.

To achieve the goal, the following objectives were set:

– to develop a hierarchical model for task allocation and resource management in IoT systems based on osmotic computing;

– to develop a task allocation method using a fuzzy inference system considering task characteristics, computing resource constraints, and optimization criteria;

– to conduct an experimental investigation of the proposed method to evaluate its effectiveness under various scenarios and load conditions.

## 4. The study materials and methods of decentralized task allocation in hierarchical IoT systems

### 4.1. Features of the general IoT system architecture

This work considers a three-tier architecture of distributed IoT systems with a division into cloud, fog, and edge environments [4, 5].

The first tier, which is the lowest in the hierarchical structure, is the edge computing environment. It is located closest to data sources, which are end IoT devices, enabling it to perform preliminary processing of this data, thereby ensuring simple local computing with low latency. This tier can accommodate both simple devices (sensors and actuators) with weak computational capabilities and more complex devices (smart sensors, microservers, gateways) that can act as computing nodes for task execution.

The second tier is the fog computing environment, which includes intermediate computing resources at the level of micro- or regional data centers. Due to its extended computational capabilities, it is capable of processing larger data volumes and performing more complex operations than the edge tier, while providing lower latencies than the cloud environment. The Fog tier can simultaneously serve several Edge tier domains.

The third, highest tier in the distributed IoT system structure is the cloud environment, which

is located farthest from end devices and is built on data centers with practically unlimited resources, high computational power, and scalability. The Cloud tier allows analysis, processing, and storage of large data arrays while ensuring a high level of fault tolerance. The Cloud tier can simultaneously serve several Fog tier domains.

Interaction between computing environments is implemented as a hierarchically realized data exchange between IoT system tiers.

The use of the osmotic computing concept in the considered model is intended to provide dynamic management of computing resources in the distributed IoT system environment, which allows for achieving flexible load balancing between edge, fog, and cloud tiers, contributing to system adaptation to real-time changes (a sharp increase or decrease in the number of tasks, connection or disconnection of nodes, node failure, increased network latency, etc.). The idea of osmotic computing is inspired by the chemical osmosis process, where a solvent moves from an area with lower solute concentration to an area with higher solute concentration through a semi-permeable membrane, thereby equalizing concentration on both sides of the membrane. In osmotic computing, micro-elements (MELs) act as the solvent, which can migrate between different environments (Cloud, Fog, Edge) through a Software-Defined Membrane (SDMem) [5, 21].

MEL is a particular abstraction that describes services and data of an IoT application, which in the context of osmotic computing is considered as a graph of micro-elements (Fig. 1). There are two types of MELs: microservices (MS), which provide specific functional capabilities and can be deployed and migrated between different computing environments, and microdata (MD), which are transmitted between IoT system components and can have different representation formats [5, 21].

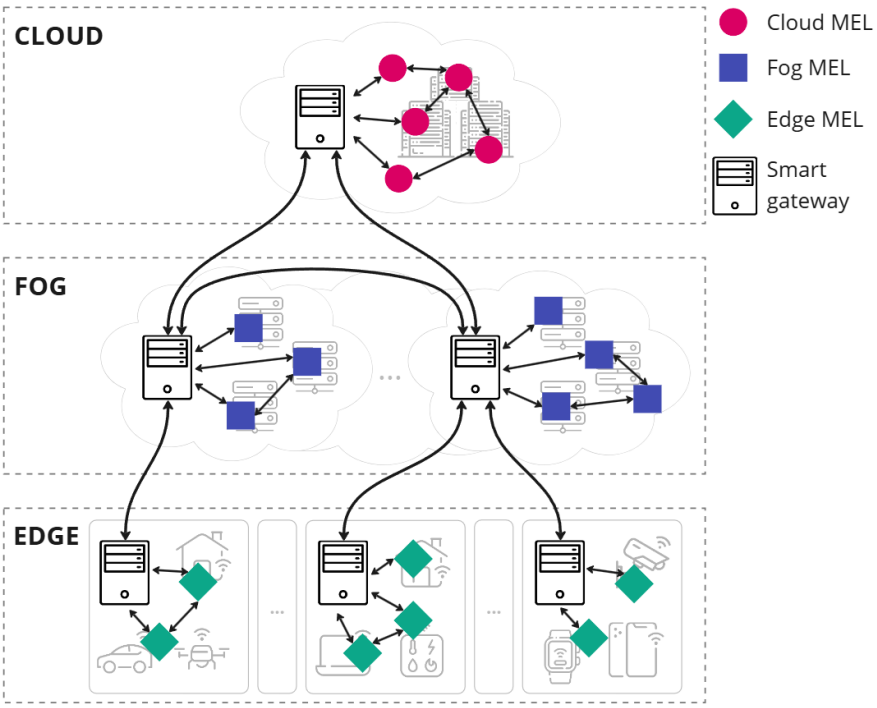


Fig. 1. General structure of an IoT application based on osmotic computing

The membrane is a specific logical software-defined layer that regulates the movement of MELs (microservices or microdata) between cloud, fog, and edge environments according to various MEL management policies, which may include computing resource availability, quality of service (QoS) requirements, current system state, security and privacy policies, etc. The membrane can be implemented as a separate software module deployed on a gateway or be part of other services or subsystems; for example, orchestration systems (such as Kubernetes) serve as tools for implementing the membrane principle – they allow automatic distribution and transfer of containerized microservices between nodes, responding to system changes [5, 21].

### 4.2. General model of task allocation and resource management in IoT systems

In the context of the considered IoT system architecture based on osmotic computing, two main management objects are distinguished: tasks and computing resources.

A task is a request for execution of a particular process. Tasks can be of two types: internal and external. Internal tasks are generated by IoT devices, microservices, or services within the system and can be formed at any hierarchy tier. Internal tasks include data processing and transmission from sensors, information aggregation, execution of business functions, and other in-system operations. External tasks originate from users or external services to interact with the system via an API, which is usually deployed in the cloud. Examples of external tasks include requests for analytical report generation from users, control commands from operators, or data collection for statistics building through integration with third-party systems.

Computing resources comprise infrastructure nodes of different tiers (edge, fog, cloud) that provide an environment for deploying, executing, and scaling MELs. They are characterized by limited computational power, energy capabilities, latency, bandwidth, and other parameters. Resource management involves monitoring node availability, planning and balancing the load, and releasing resources.

MELs should be noted separately as an additional management object, which is a unit of task execution deployed within available computing resources. MEL management includes deployment in cloud, fog, or edge environments, migration between different environments or within the same environment, as well as deletion or unloading when resources are released.

One or more MELs can be deployed on a computing node depending on available capabilities. Each MEL, in turn, can execute one or several tasks.

The considered architecture of the distributed IoT system implements a three-level hierarchical model of task and computing resource management, which covers the edge, fog, and cloud tiers (Fig. 2). Each tier contains computing nodes on which MELs are deployed, and a management system that, based on analysis of resource state monitoring data, makes decisions on the placement, migration, and removal of MELs and performs task distribution. The lower-tier management system is subordinate to the higher-tier management system in the hierarchy.

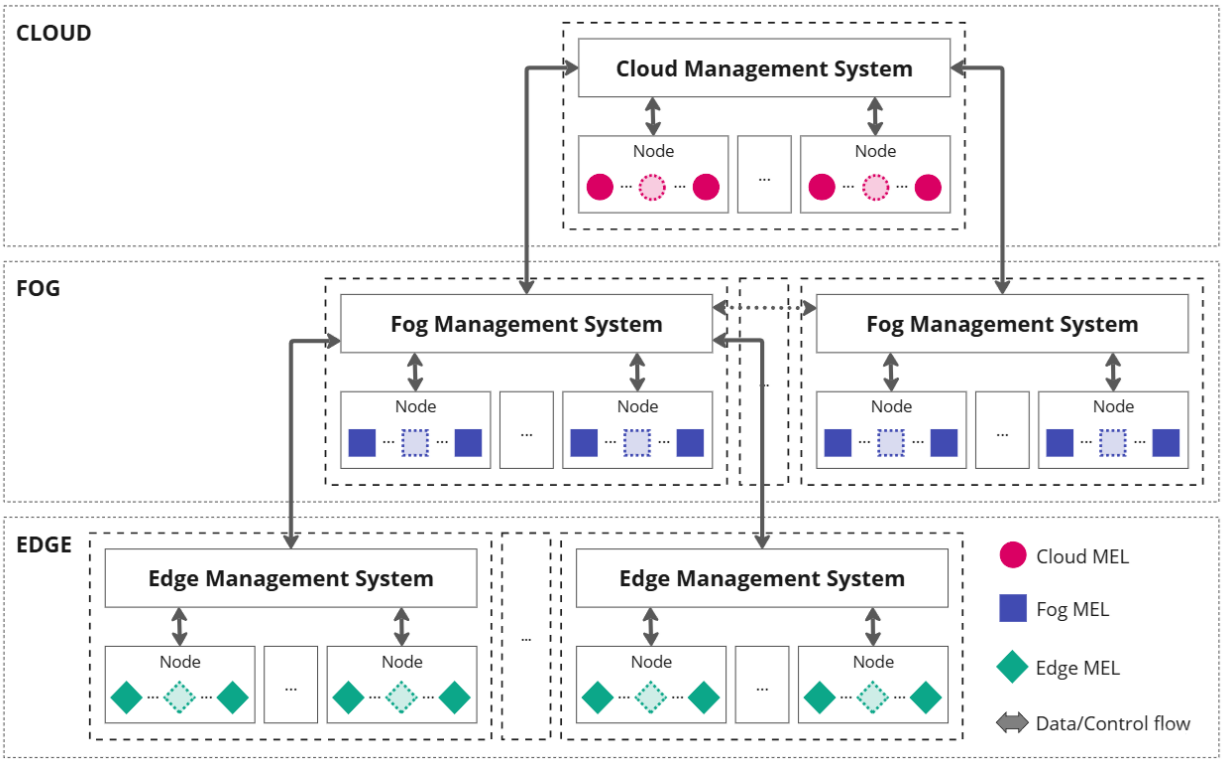


Fig. 2. General model of task allocation and resource management in IoT systems

The Edge management system manages computing nodes that interact with end devices in IoT systems and ensures task execution near data sources. The Fog management system coordinates one or several edge domains – groups of computing nodes subordinate to a single management system – and computing nodes within its domain, providing intermediate decision-making capabilities. Horizontal interaction between Fog management systems of different domains is possible, allowing task redistribution between these domains and ensuring the operation of the IoT system even when communication with the central management tier is disrupted, thereby improving overall system reliability and adaptability. The central management system, located at the cloud tier, has information about the overall IoT system state, coordinates Fog domains, and ensures coordinated operation of all tiers, making strategic decisions regarding task allocation and computing resource management.

The considered management model implements a combination of centralized and decentralized approaches to management. The central management system performs global management and decision-making. At the same time, decentralized management is carried out by management systems of corresponding tiers, which make local decisions and manage within their domain. The combination of these approaches allows to increase the fault tolerance, scalability, and adaptability of the system.

The current state of the IoT system is determined by a monitoring subsystem implemented based on an agent-based approach (Fig. 3).

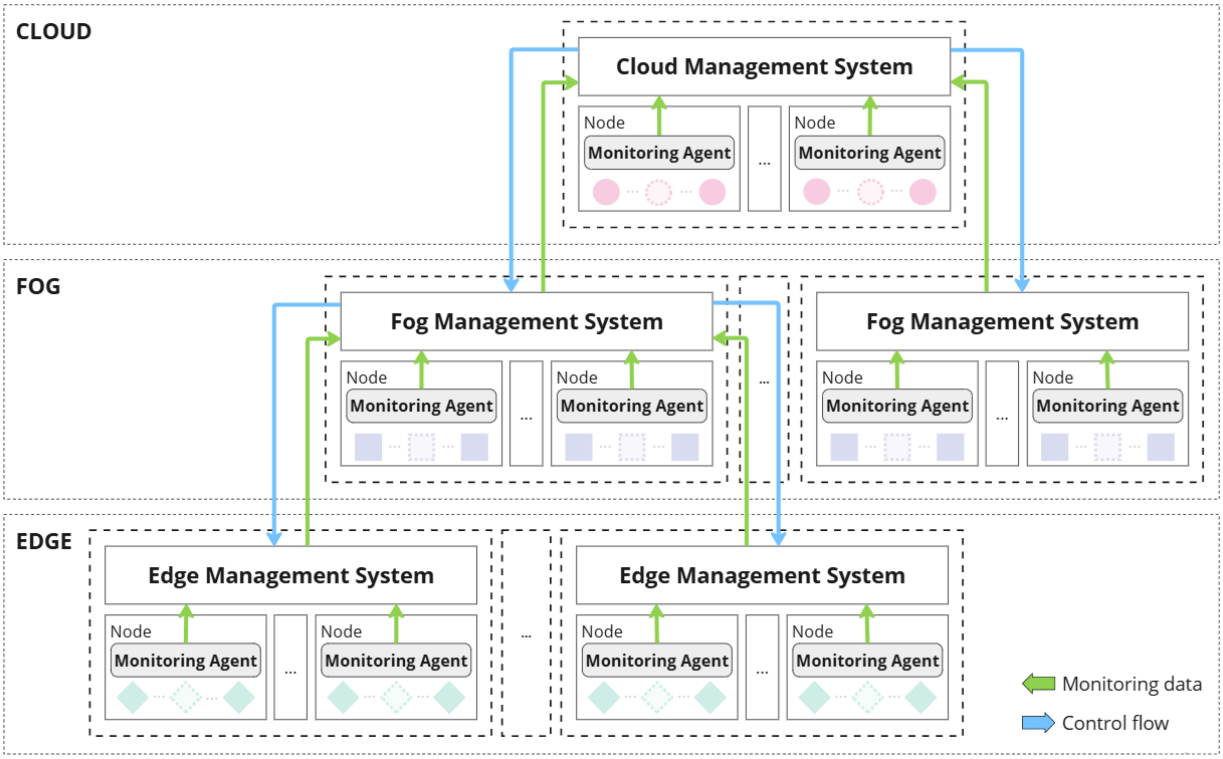


Fig. 3. General structure of the IoT system monitoring subsystem

A monitoring agent is deployed on each computing node, which collects necessary metrics (CPU/GPU utilization, memory, energy consumption, network bandwidth, etc.). The local management system receives information about the state of each computing node and sends it to the higher-tier management system, performing preliminary processing and data aggregation. Monitoring agents can send metrics directly to the management system, or the local management system can poll agents at regular intervals. The central management system has complete information about the current state of the entire IoT system, analyzes it, and, if necessary, regulates MEL management policies.

### 4.3. Formalization of the task allocation problem in IoT systems

In the considered three-tier architecture of a distributed IoT system based on osmotic computing, the environment for executing tasks $T_i \in \mathrm{T}$, $i = \overline{1, I}$ is MELs, which are deployed on computing node $N_j \in \mathrm{N}_L$, $j = \overline{1, J_L}$, where $\mathrm{N}_L$ – is the set of computing nodes at tier $L \in \mathrm{L}$, $\mathrm{L} = \{Edge, Fog, Cloud\}$, $J_L$ – is the number of available nodes at tier $L$.

Each task $T_i \in \mathrm{T}$ is described by a vector of parameters $P_i = \{p_i^k \mid k = \overline{1, K}\}$, where $p_i^k$ – value of the $k$-th parameter of the $i$-th task. Task parameters include priority, latency requirements, computational complexity, etc. The number and types of parameters may vary depending on the specific requirements and features of an IoT system.

Each computing node $N_j \in \mathrm{N}_L$ has resources $R_j \in \mathrm{R}_L$, where $\mathrm{R}_L$ – total resources at tier $L \in \mathrm{L}$. Within the same tier, computing node resources may be heterogeneous $R_j \neq R_m$, $R_j \in \mathrm{R}_L$, $R_m \in \mathrm{R}_L$ for some $j \neq m$, where $j = \overline{1, J_L}$, $m = \overline{1, J_L}$. The resources $R_j \in \mathrm{R}_L$ of computing node $N_j \in \mathrm{N}_L$ are characterized by a vector of parameters $Q_j = \{q_j^m \mid m = \overline{1, M}\}$, where $q_j^m$ – the value of the $m$-th parameter of the $j$-th computing node. Computing node parameters include memory, computational power, network bandwidth, energy consumption, etc.

The task allocation problem consists of determining the optimal placement of tasks $\mathrm{T}$ on computing nodes $\mathrm{N} = \bigcup_{L \in \mathrm{L}} \mathrm{N}_L$:

$$X = \{x_{i,j,L}\},\ x_{i,j,L} \in \{0,1\},\ i = \overline{1, I},\ j = \overline{1, J_L}, \tag{1}$$

where $x_{i,j,L} = 1$, if task $T_i \in \mathrm{T}$ is assigned to computing node $N_j \in \mathrm{N}_L$ at tier $L \in \mathrm{L}$, and $x_{i,j,L} = 0$ otherwise, such that task execution delays are minimal while ensuring rational resource utilization.

The total latency of task $T_i \in \mathrm{T}$ can be defined as

$$F_{i,j,L}^l = F_{i,j,L}^t + F_{i,j,L}^e + F_{i,j,L}^r, \tag{2}$$

where $F_{i,j,L}^t$ – the transmission time of task $T_i \in \mathrm{T}$ from the initialization source to computing node $N_j \in \mathrm{N}_L$, $F_{i,j,L}^e$ – the execution time of $T_i \in \mathrm{T}$ on the node $N_j \in \mathrm{N}_L$, $F_{i,j,L}^r$ – the transmission time of the result back to the initialization source. Since the task initialization tier and its execution tier may differ, we define $F_{i,j,L}^t$ and $F_{i,j,L}^r$ as

$$F_{i,j,L}^t = F_{i,j,L}^{t,L} + F_{i,j,L}^{t,N_j},\ F_{i,j,L}^r = F_{i,j,L}^{r,L} + F_{i,j,L}^{r,N_j}, \tag{3}$$

where $F_{i,j,L}^{t,L}$, $F_{i,j,L}^{r,L}$ – the data transmission time to/from tier $L$, respectively, $F_{i,j,L}^{t,N_j}$, $F_{i,j,L}^{r,N_j}$ – the data transmission time to/from computing node $N_j \in \mathrm{N}_L$ at tier $L$, respectively. $F_{i,j,L}^{r,L}$, if the task is assigned to a computing node at the same tier where it was initialized. $F_{i,j,L}^{t,L}$ if the task is assigned to a computing node at the same tier where it was initialized, without being redirected to another tier.

The total latency for all tasks is defined as:

$$F_{total}^l = \sum_{i=1}^{I} \sum_{L \in \mathrm{L}} \sum_{j=1}^{J_L} x_{i,j}^L \cdot F_{i,j,L}^l, \tag{4}$$

where $F_{i,j,L}^l$ – the total latency of task $T_i \in \mathrm{T}$ on node $N_j \in \mathrm{N}_L$ of tier $L$.

The rational use of computing resources is determined by the degree of efficiency of their use and the balance of load distribution.

The resource utilization efficiency for node $N_j \in \mathrm{N}_L$ at tier $L$ is defined as

$$F_{j,L}^{ru} = \frac{\sum_{i=1}^{I} x_{i,j,L} \cdot r_i}{R_j}, \tag{5}$$

where $R_j$ – the total resources of node $N_j$, $r_i$ – the resource requirements of task $T_i$.

The resource utilization efficiency at tier $L$ is defined as the weighted average of node utilization rates:

$$F_L^{ru} = \frac{\sum_{j=1}^{J_L} F_{j,L}^{ru} \cdot R_j}{\sum_{j=1}^{J_L} R_j} = \frac{\sum_{j=1}^{J_L} \sum_{i=1}^{I} x_{i,j,L} \cdot r_i}{\sum_{j=1}^{J_L} R_j}, \tag{6}$$

where $J_L$ – the number of available nodes at tier $L$.

Similarly to (6), we define resource utilization efficiency for the entire IoT system:

$$F_{total}^{ru} = \frac{\sum_{L \in \mathrm{L}} F_L^{ru} \cdot \mathrm{R}_L}{\sum_{L \in \mathrm{L}} \mathrm{R}_L} = \frac{\sum_{L \in \mathrm{L}} \sum_{j=1}^{J_L} \sum_{i=1}^{I} x_{i,j,L} \cdot r_i}{\sum_{L \in \mathrm{L}} \sum_{j=1}^{J_L} R_j}, \tag{7}$$

where $\mathrm{R}_L = \sum_{j=1}^{J_L} R_j$ – the total resources on tier $L \in \mathrm{L}$.

Since computing nodes $N_j \in \mathrm{N}_L$ within the same tier $L$ may have different computing capacities, normalized metrics that account for the relative load of each node should be used to assess load balancing. The overall load balance evaluation at tier $L$ among heterogeneous nodes is determined using Jain fairness index [22]:

$$F_L^{lb} = \frac{\left( \sum_{j=1}^{J_L} F_{j.L}^{ru} \right)^2}{J_L \cdot \sum_{j=1}^{J_L} \left( F_{j.L}^{ru} \right)^2}, \tag{8}$$

where $F_L^{lb} \in \left[ \frac{1}{J_L}, 1 \right]$, $J_L$ – the number of available nodes at tier $L$. A value of $F_L^{lb}$ close to 1 indicates uniform load balancing at tier $L$, and vice versa, close to $\frac{1}{J_L}$ indicates a strong imbalance.

Finally, task allocation in an IoT system can be formalized as a multi-criteria optimization problem of finding task allocation $X$, that ensures:

$$F_{total}^{l} \to \min, \; F_{total}^{ru} \to \max, \; F_{total}^{lb} \to \max \tag{9}$$

subject to the following constraints:

– each task can be assigned to only one node: $\sum_{L \in \mathrm{L}} \sum_{j=1}^{J_L} x_{i,j,L} = 1$, $x_{i,j,L} \in \{0,1\}$, $\forall i \in I$, $\forall j \in J_L$, $\forall L \in \mathrm{L}$;

– the total load on a node must not exceed its computing resources: $\sum_{i=1}^{I} x_{i,j,L} \cdot r_i \le R_j$, $\forall j \in J_L$, $\forall L \in \mathrm{L}$.

### 4.4. Decentralized task allocation method using fuzzy logic

In the considered hierarchical management model in the IoT system shown in Figure 2, a task allocation method is proposed that considers both the characteristics and requirements of tasks and the state of computing resources to determine the rational execution environment. In general, tasks are allocated according to Algorithm 1.

**Algorithm 1.** Task allocation for hierarchical IoT systems

**Input:** Task $T_i$ with parameters $P_i$, set of nodes $\mathrm{N} = \bigcup_{L \in \mathrm{L}} \mathrm{N}_L$ with resources $\mathrm{R} = \bigcup_{L \in \mathrm{L}} \mathrm{R}_L$, $\mathrm{L} = \{Edge, Fog, Cloud\}$.

**Output:** Assignment of task $T_i$ to a node or queue/reject the task.

$L \leftarrow$ `tier where` $T_i$ `was initialized`
$\mathrm{L}_{available} \leftarrow \mathrm{L}$
**while** $L \neq null$ **do**
  $D \leftarrow$ `selectSuitableDirection(`$P_i$`)`
  **if** $D = current$ **then**
    $\mathrm{N}_{suitable,L} \leftarrow \{ N_{j,L} \in \mathrm{N}_L \mid R_{j,L} \geq$ `requiredResources(`$P_i$`)` $\}$
    **if** $\mathrm{N}_{suitable,L} \neq \varnothing$
      $N_{target,L} \leftarrow$ `selectSuitableNode(`$P_i$`,` $\mathrm{N}_{suitable,L}$`)`
      `assignTask(`$T_i$`,` $N_{target,L}$`)`
      **return**
    **end if**
  **end if**
  **if** $D = current$ **then**
    $D \leftarrow up$
    $\mathrm{L}_{available} \leftarrow \mathrm{L}_{available} \setminus \{L\}$
  **end if**
  **if** $D \neq current$ **then**
    $L \leftarrow$ `selectNextLevel(`$L$`,` $D$`)`
    **if** $L \notin \mathrm{L}_{available}$ **then**
      $L \leftarrow null$
    **end if**
  **end if**
**end while**
`queueOrRejectTask(`$T_i$`)`

The task $T_i \in \mathrm{T}$ is registered in the local management system at the tier $L \in \mathrm{L}$ where it is initiated. In the first step, the local management system determines the task allocation direction $D_i$. It evaluates the feasibility of executing the task at the current tier $L$ or transferring it to another tier based on task characteristics $P_i = \{p_i^k \mid k = \overline{1,K}\}$. If the current tier $L$ meets the requirements for execution of the task $T_i$, the allocation direction $D_i$ indicates the current tier $L$. Then, the management system checks the availability of nodes $\mathrm{N}_{suitable,L} \subseteq \mathrm{N}_L$ at the tier $L$ with the necessary resources $R_j \in \mathrm{R}_L$ for task computation on node $N_{j,L} \in \mathrm{N}_{suitable,L}$. In the next step, the local management system determines the optimal node $N_{target,L} \in \mathrm{N}_{suitable,L}$ for placing the task $T_i$ among

the available computing nodes $\mathrm{N}_{suitable,L}$.

If the current tier $L$ meets the requirements of the task $T_i$, but does not contain available nodes $\mathrm{N}_{suitable,L} = \varnothing$ with the necessary computing resources for executing the task $T_i$, then the task allocation direction $D_i$ changes to upward, and the current tier $L$ is excluded from the set of permissible tiers $\mathrm{L}_{available} \leftarrow \mathrm{L}_{available} \setminus \{L\}$. The change of allocation direction $D_i$ to upward is justified by the fact that at the upper hierarchy tier, the probability of having a node with the necessary computing resources is higher than at the lower tier.

If the current tier $L$ does not meet the requirements of the task $T_i$, i.e., the task allocation direction $D_i$ indicates redirecting the task to a higher or lower tier. Then the management system determines the next tier for task transfer based on the current tier $L$ value and the allocation direction $D_i$. If the new tier belongs to the set of available tiers $\mathrm{L}_{available}$, then the task is redirected to the determined tier, after which the analysis process is repeated. If the new tier does not belong to the set of available tiers $\mathrm{L}_{available}$, then the task $T_i$ is placed in a queue or rejected depending on the policies implemented in the management system.

To determine the task allocation direction `selectSuitableDirection(`$P_i$`)` and select the computing node for its execution `selectSuitableNode(`$P_i$, $\mathrm{N}_{suitable,L}$`)`, Mamdani fuzzy inference systems are used [11–13]. This approach allows for formalizing the decision-making process under uncertain conditions, which are associated with dynamic changes in task characteristics, computing resource availability, and load in the IoT system.

In general, a fuzzy logic inference system consists of the following stages (Fig. 4) [11–13]:

– Fuzzification – converting crisp input values into fuzzy sets by determining their degree of membership to corresponding linguistic terms.

– Rule base application – applying a set of fuzzy “if-then” rules that describe dependencies between input and output parameters.

– Fuzzy logical inference – determining the degree of truth of each rule using fuzzy operators.

– Defuzzification – converting the obtained fuzzy results into a crisp value using defuzzification methods.

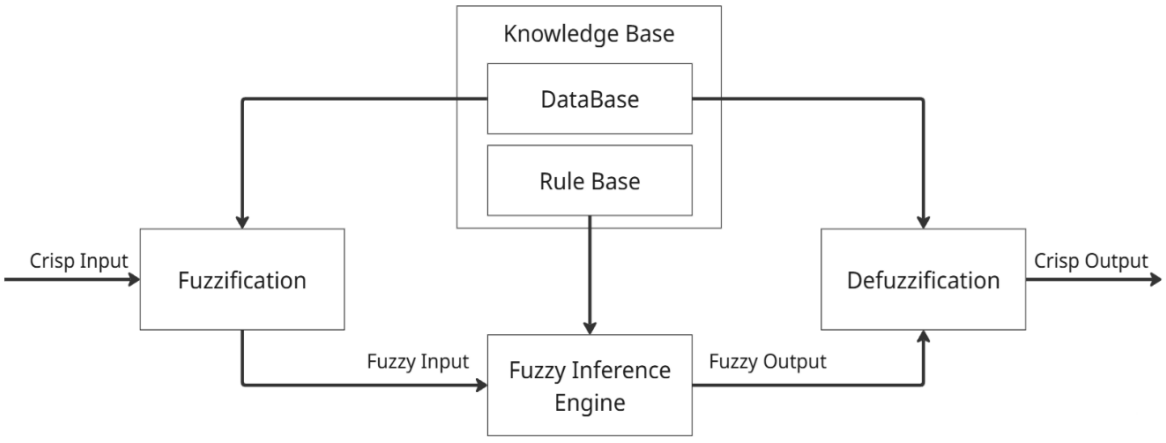


Fig. 4. Fuzzy logic inference system

### 4.4.1. Fuzzy inference system for determining allocation direction

The rational allocation direction $D_i$ of the task $T_i \in \mathrm{T}$ is determined using a fuzzy logic inference system based on task parameters $P_i = \{p_i^k \mid k = \overline{1,K}\}$ (Fig. 5).

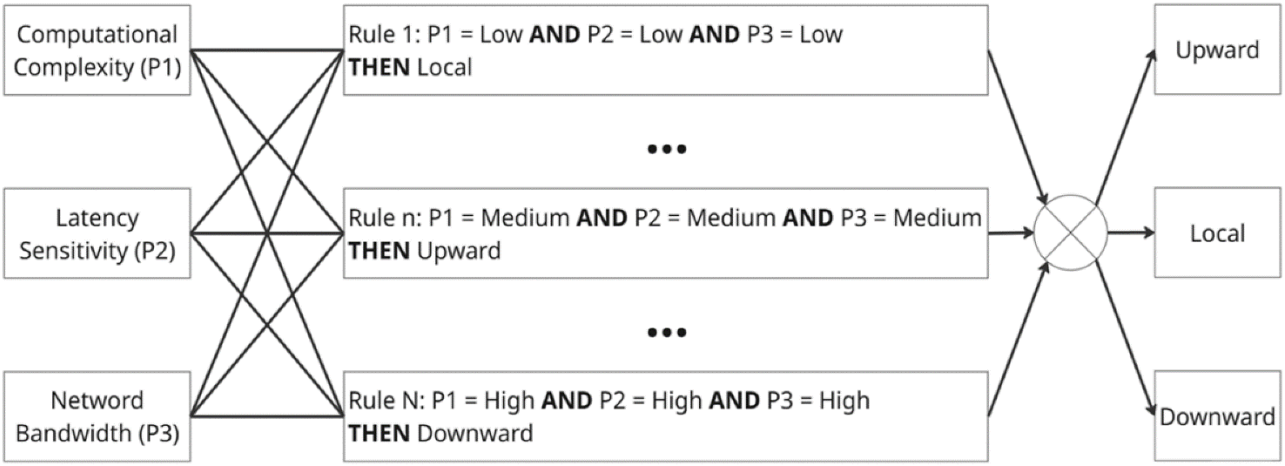


Fig. 5. Determining the direction of task allocation using fuzzy logic

According to the task allocation criteria, the following task parameters are considered as input parameters of the fuzzy inference system: computational complexity, latency sensitivity, and network bandwidth requirements.

The computational complexity $P_i^{cc}$ of a task determines the computing power requirements of the node for its execution. Tasks with high computational complexity require more powerful resources, available at higher hierarchy tiers (Fog, Cloud), while tasks with low complexity can be effectively executed on lower-tier devices (Edge).

Latency sensitivity $P_i^{ls}$ characterizes the degree to which a task is critical to its execution speed. Tasks with high latency sensitivity need to be executed closer to the data source to minimize overall latency. Conversely, tasks with low latency sensitivity can be executed on remote computing nodes without a significant impact on quality of service.

Network bandwidth requirements $P_i^{nb}$ determine the minimum amount of data that needs to be transmitted through the network per unit of time for effective task execution. Tasks with high bandwidth requirements can create a significant network load when transmitted to a remote tier; therefore, such tasks are more efficiently executed locally or at intermediate tiers to reduce network traffic.

For the described task parameters $P_i^{cc}$, $P_i^{ls}$, and $P_i^{nb}$ triangular membership functions are used, which determine the degree of parameter value membership to linguistic terms “low”, “medium”, and “high”. Triangular membership functions are the most common, as they provide computational simplicity and smooth transitions between terms.

The output parameter of the fuzzy inference system is the allocation direction $D_i$ of the task $T_i$, the fuzzy value of which is described by one of three linguistic terms:

– “current” – the task remains at the current tier – accepted for tasks if the tier characteristics meet task parameters and the tier load allows its execution;

– “upward” – the task is transferred to a higher hierarchy tier – accepted for tasks with high computational complexity or when the current tier is overloaded;

– “downward” – the task is transferred to a lower hierarchy tier – accepted for tasks with low computational complexity and/or high latency sensitivity.

For Cloud and Edge tiers, the task allocation direction can have only two values: “current” and “downward” for Cloud, or “current” and “upward” for Edge.

Based on input and output linguistic terms, a fuzzy rule base is formed, which contains a set of “if-then” rules, each determining the correspondence between a combination of input linguistic terms and a single output linguistic decision. Each tier contains its own fuzzy rule base, which differs from rule bases of other tiers. Tables 1, 2, and 3 present part of the rules for Edge, Fog, and Cloud tiers, respectively.

Table 1. Fuzzy rules for task allocation at the Edge tier

| **Input parameters** | | | **Output parameters** |
|---|---|---|---|
| **Computational complexity** $P_i^{cc}$ | **Latency sensitivity** $P_i^{ls}$ | **Network bandwidth** $P_i^{nb}$ | **Allocation direction** $D_i$ |
| Low | High | Low | Current |
| Low | High | Medium | Current |
| Low | Medium | Low | Current |
| Medium | High | Low | Current |
| Medium | Medium | High | Current |
| Medium | Low | Medium | Upward |
| High | High | Low | Upward |
| High | Medium | Medium | Upward |
| High | Low | Low | Upward |
| … | … | … | … |

Table 2. Fuzzy rules for task allocation at the Fog tier

| **Input parameters** | | | **Output parameters** |
|---|---|---|---|
| **Computational complexity** $P_i^{cc}$ | **Latency sensitivity** $P_i^{ls}$ | **Network bandwidth** $P_i^{nb}$ | **Allocation direction** $D_i$ |
| Low | High | Low | Downward |
| Low | High | Medium | Downward |
| Low | Medium | Low | Current |
| Low | Low | Medium | Current |
| Medium | High | Low | Current |
| Medium | Medium | Medium | Current |
| Medium | Low | Low | Upward |
| High | High | High | Current |
| High | Medium | Medium | Current |
| High | Low | Low | Upward |
| … | … | … | … |

Table 3. Fuzzy rules for task allocation at the Cloud tier

| **Input parameters** | | | **Output parameters** |
|---|---|---|---|
| **Computational complexity** $P_i^{cc}$ | **Latency sensitivity** $P_i^{ls}$ | **Network bandwidth** $P_i^{nb}$ | **Allocation direction** $D_i$ |
| Low | High | Low | Downward |
| Low | High | Medium | Downward |
| Low | Medium | Low | Downward |
| Low | Low | Medium | Downward |
| Medium | High | Low | Downward |
| Medium | Medium | Medium | Current |
| Medium | Low | Low | Current |
| High | High | Low | Current |
| High | Medium | Medium | Current |
| High | Low | Medium | Current |
| … | … | … | … |

Using the rule base, the fuzzy inference engine determines the fuzzy value for the allocation direction $D_i$ of the task $T_i$ based on fuzzy values of task parameters. The final step involves defuzzification of the fuzzy value using the center of gravity method to obtain a crisp value for the task allocation direction.

### 4.4.2. Fuzzy logic inference system for determining a computing node

After determining the rational allocation direction for task $T_i \in \mathrm{T}$ with its subsequent placement at the corresponding tier $L$, the local management system of the tier selects the optimal node $N_{target,L} \in \mathrm{N}_{suitable,L}$ for task execution. The determination of the target node $N_{target,L}$ is performed using a fuzzy inference system (Fig. 6).

The input parameters of the fuzzy inference system are the formalized optimization criteria of the allocation problem: task latency $Q_{i,j}^{l}$, resource utilization efficiency $Q_{i,j}^{ru}$, and load balancing $Q_{i,j}^{lb}$ at the tier.

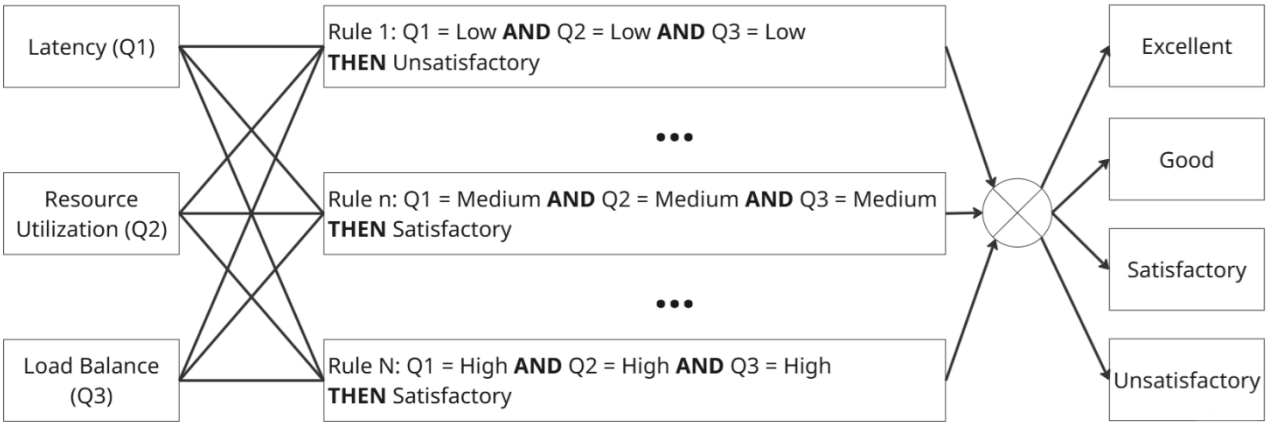


Fig. 6. Determining a computing node for task execution using fuzzy logic

For each computing node $N_{j,L} \in \mathrm{N}_{suitable,L}$, the total latency of the task $T_i$ is calculated according to (2).

The resource utilization efficiency for a node $N_{j,L}$ after assigning the task $T_i$ to it according to (6) is calculated as

$$Q_{i,j}^{ru} = \frac{r_i + \sum_{j=1}^{J_L} \sum_{k=1}^{K} r_k}{\sum_{j=1}^{J_L} R_j}, \tag{13}$$

where $r_i$ – resource requirements of the task $T_i$, $r_k$ – resource requirements of task $T_k \in \mathrm{T}_{existing}$, $k = \overline{1,K}$, $K \le I$, $\mathrm{T}_{existing}$ – set of tasks already executing on the node $N_{j,L}$, $J_L$ – number of available nodes at tier $L$.

The load balance at the tier $L$ after assigning the task $T_i$ to the node $N_{j,L}$ is calculated using Jain fairness index according to formula (8).

For each of the three input parameters $Q_{i,j}^{l}$, $Q_{i,j}^{ru}$, and $Q_{i,j}^{lb}$ triangular membership functions are used, which determine the degree of membership of normalized values to the linguistic terms “low”, “medium”, and “high”.

The output parameter of the system is the node suitability score $S_{i,j}$, which characterizes the degree of correspondence of node $N_{j,L} \in \mathrm{N}_{suitable,L}$ for executing task $T_i$. The fuzzy value of the score is represented by one of four linguistic terms:

– "unsatisfactory" – the node is not suitable for task execution,
– "satisfactory" – the node can execute the task with minimal quality,
– "good" – the node can execute the task with good quality,
– "excellent" – the node is best suited for task execution.

Table 4 presents a set of fuzzy rules used by the fuzzy logic inference engine to determine the fuzzy value of score $S_{i,j}$ for the correspondence of node $N_{j,L}$ for executing task $T_i$. The final step is defuzzification using the center of gravity method to obtain a crisp value.

After evaluating the suitability rating of nodes from the set $\mathrm{N}_{suitable,L}$, task $T_i$ is assigned to the node $N_{target,L}$ with the maximum rating value. If several nodes have the same maximum rating value, the node with the lower load value is selected to ensure a uniform load balance.

Table 4. Fuzzy rules for determining node suitability score for task execution

| **Input parameters** | | | **Output parameters** |
|---|---|---|---|
| **Task latency** $Q_{i,j}^{l}$ | **Resource utilization** $Q_{i,j}^{ru}$ | **Load balance** $Q_{i,j}^{lb}$ | **Node score** $S_{i,j}$ |
| Low | Low | Low | Unsatisfactory |
| Low | Low | Medium | Satisfactory |
| Low | Medium | Medium | Good |
| Medium | Low | Low | Unsatisfactory |
| Medium | Low | High | Satisfactory |
| Medium | Medium | High | Good |
| High | Low | High | Unsatisfactory |
| High | Medium | High | Satisfactory |
| High | High | Low | Unsatisfactory |
| … | … | … | … |

The proposed fuzzy logic inference system allows for comprehensive consideration of optimization criteria, ensuring decision-making in conditions of incomplete or inaccurate information about the state of the level.

## 5. Results of the investigation of decentralized task allocation method in hierarchical IoT systems

### 5.1 Experimental setup

For modeling and investigating the proposed decentralized task allocation method, we used the iFogSim2 simulator [23–25], which is built on the CloudSim framework [26, 27]. iFogSim2 is an open-source toolkit for modeling and simulating task allocation methods and resource management in multi-tier IoT systems, supporting edge and fog environments [23–25]. The open-source Java library jFuzzy was used to implement fuzzy inference systems and integrate them with iFogSim2. The membership functions of input and output parameters of the fuzzy inference systems have a triangular shape. The fuzzy inference system for determining task allocation direction has its rule base with 27 rules at each hierarchy tier, while the fuzzy inference system for evaluating node suitability rating uses a unified rule base that also contains 27 rules.

The characteristics of the computing resources, such as processor capacity, memory capacity, and bandwidth, were selected based on real devices that can be deployed at corresponding tiers of the

IoT system hierarchy. Table 5 presents examples of physical devices with corresponding characteristics and their deployment tiers.

For conducting the simulation, the following configuration of computing nodes was selected: 6 nodes at the edge tier, 3 at the fog tier, and 2 at the cloud tier. The characteristics of computing nodes at corresponding tiers are presented in Table 6.

Table 5. Example of general characteristics of computing nodes

| Tier | Type | Processing power, MIPS | Memory | Bandwidth | Examples |
|---|---|---|---|---|---|
| Edge | Microcontrollers | 16–600 | 2–520 MB | 10–50 Mbps | Arduino Uno, ESP32 |
| | Single-board computers | 6000–25000 | 512–8192 MB | 100–1000 Mbps | Raspberry Pi 4B, BeagleBone |
| | Edge-gateways | 8000–20000 | 4–8 GB | 1000–2500 Mbps | Intel NUC, Advantech ARK, Moxa UC |
| Fog | Industrial PCs | 18000–30000 | 8–16 GB | 1–5 Gbps | Siemens SIMATIC, Cisco Iox |
| | Microservers | 18000–35000 | 8–32 GB | 1–10 Gbps | Intel NUC Pro, HPE MicroServer |
| Cloud | Small VM | 15000–85000 | 1–16 GB | 5–10 Gbps | AWS t3/m5, Azure Standard |
| | Medium VM | 85000–150000 | 32–64 GB | 10–15 Gbps | AWS c5/m5.xlarge, Azure F-series |
| | Large VM | 150000–500000 | 64–128 GB | 10–25 Gbps | AWS p3/p4, Azure NC-series |

Table 6. Characteristics of computing nodes in simulation

| Tier | Processing power, MIPS | Bandwidth, Mbps | Latency between node and local management system, ms |
|---|---|---|---|
| Edge | 300 | 20000 | 10 |
| | 600 | 30000 | 10 |
| | 12000 | 40000 | 10 |
| | 15000 | 50000 | 10 |
| | 18000 | 45000 | 10 |
| | 14000 | 35000 | 10 |
| Fog | 24000 | 800000 | 50 |
| | 26500 | 600000 | 50 |
| | 50000 | 700000 | 50 |
| Cloud | 120000 | 1800000 | 100 |
| | 325000 | 2000000 | 100 |

Data transmission latency between tiers is:

– Edge-Fog – 10 ms,

– Fog-Cloud – 50 ms.

To investigate the behavior of IoT systems under different conditions, 13 sets of scenarios were modeled, including varying numbers of tasks, different task generation intervals, and diverse task characteristics. Tasks are generated at each tier in equal numbers. The number and characteristics of computing nodes remain unchanged for all scenarios. Table 7 presents the main parameters of all scenarios.

Table 7. Simulation scenario parameters

| Scenario | Number of tasks at each tier | Total number of tasks | Tier | Tasks interval, ms | Task characteristics | | |
|---|---|---|---|---|---|---|---|
| | | | | | Computational complexity, MIPS | Latency sensitivity, p.u. | Size, b |
| Basic | 50/ 250/ 500 | 150/ 750/ 1500 | Edge | 500 | 150–2000 | 0.7–1.0 | 500–8000 |
| Basic | 50/ 250/ 500 | 150/ 750/ 1500 | Fog | 800 | 1000–8000 | 0.2–0.8 | 2000–25000 |
| | | | Cloud | 1000 | 15000–100000 | 0.1–0.6 | 10000–100000 |
| Low | 50/ 250/ 500 | 150/ 750/ 1500 | Edge | 800 | 150–2000 | 0.7–1.0 | 500–8000 |
| | | | Fog | 1200 | 1000–8000 | 0.2–0.8 | 2000–25000 |
| | | | Cloud | 1500 | 15000–100000 | 0.1–0.6 | 10000–100000 |
| High | 50/ 250/ 500 | 150/ 750/ 1500 | Edge | 100 | 150–2000 | 0.7–1.0 | 500–8000 |
| | | | Fog | 200 | 1000–8000 | 0.2–0.8 | 2000–25000 |
| | | | Cloud | 200 | 15000–100000 | 0.1–0.6 | 10000-100000 |
| Compute-Intensive | 100 | 300 | Edge | 500 | 150–8000 | 0.1–0.6 | 500–600 |
| | | | Fog | 800 | 1000–25000 | 0.1–0.5 | 1000–2000 |
| | | | Cloud | 1000 | 15000–150000 | 0.1–0.4 | 2000–5000 |
| Latency-Intensive | 100 | 300 | Edge | 500 | 150–1000 | 0.6–1.0 | 500–600 |
| | | | Fog | 800 | 1000–2500 | 0.4–1.0 | 1000–2000 |
| | | | Cloud | 1000 | 1000–2500 | 0.5–1.0 | 2000–5000 |
| Bandwidth -Intensive | 100 | 300 | Edge | 500 | 150–1000 | 0.2–0.5 | 5000–50000 |
| | | | Fog | 800 | 1000–2500 | 0.1–0.4 | 15000–100000 |
| | | | Cloud | 1000 | 1000–2500 | 0.1–0.4 | 50000–500000 |

To evaluate and compare the results of the task allocation method in each scenario, the corresponding metrics of tasks and computing nodes were collected.

Task evaluation metrics:

– Task assignment latency – time from task initialization to task assignment to a computing node. This metric accounts for all routing latency, including latency between computing nodes and the local management system, as well as transmission latency between tiers;

– Task computation latency – time from task assignment to a computing node to task completion on that node;

– Task response latency – time from task completion on a computing node to result return to the task initiator node;

– Total task latency – includes assignment, computation, and response latencies;

– Task initialization tier and task assignment tier.

Computing node evaluation metrics:

– Node load at a specific time;

– Entire tier load at a specific time;

– Tier load balancing index at a specific time.

### 5.2 Basic scenario

In the basic scenario, 50 tasks were initiated at each tier with moderate values of computational complexity, latency sensitivity, and data volume. Generation intervals were 500 ms, 800 ms, and 1000 ms for Edge, Fog, and Cloud tiers, respectively. The relationship between the number of tasks initiated at a certain tier and the number of tasks assigned to that tier for execution is shown in Figure 7.

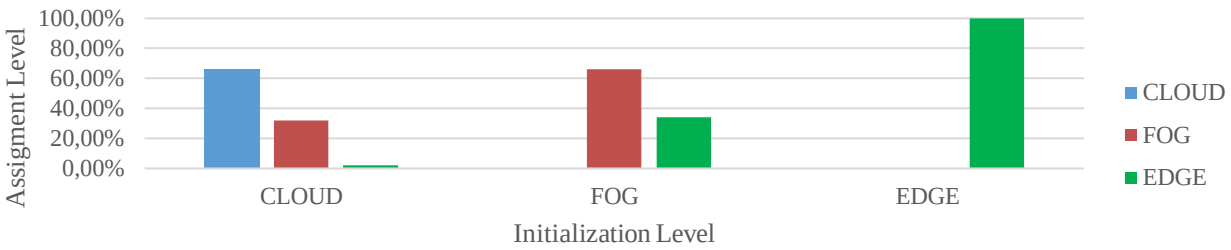


Fig. 7. Ratio of tasks initiated at tier to tasks executed at tier in Basic scenario for 50 tasks

As shown in Figure 7, all tasks initiated at the Edge tier were executed locally, while for the Fog and Cloud tiers, the share of locally executed tasks is 66%. At the same time, 34% of Fog tasks were executed on Edge, while 34% of Cloud tasks were distributed between Fog (32%) and Edge (2%). Overall, only 23% of tasks were assigned to a tier different from the initialization tier, indicating predominantly local decision-making.

The average values of assignment, computation, and response latencies for all tasks according to their initialization and assignment tiers are presented in Figure 8.

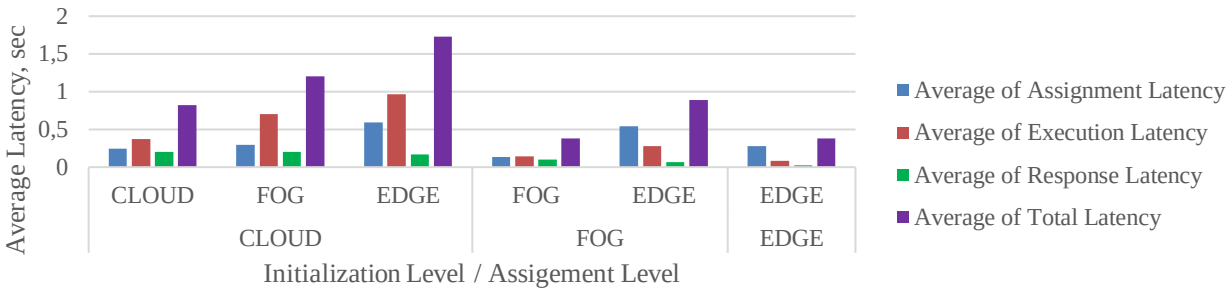


Fig. 8. Average task latency according to initialization and assignment tier in Basic scenario for 50 tasks

The lowest latency is observed with local task execution, with the lowest average task assignment latency being 130 ms for the Fog tier, and the lowest average computation time of 80 ms and response latency of 20 ms for the Edge tier. When transferring tasks from higher hierarchy tiers to lower ones, an increase in all types of latency is observed, which is related to network task routing costs, reduced computational capabilities, and device bandwidth. The total task execution latency increases by an average of 400 ms when transferring tasks to lower tiers, which may indicate the need for further research and optimization of inter-tier task routing.

The dynamics of computing node load and load balancing index for tiers and the IoT system are shown in Figures 9 and 10.

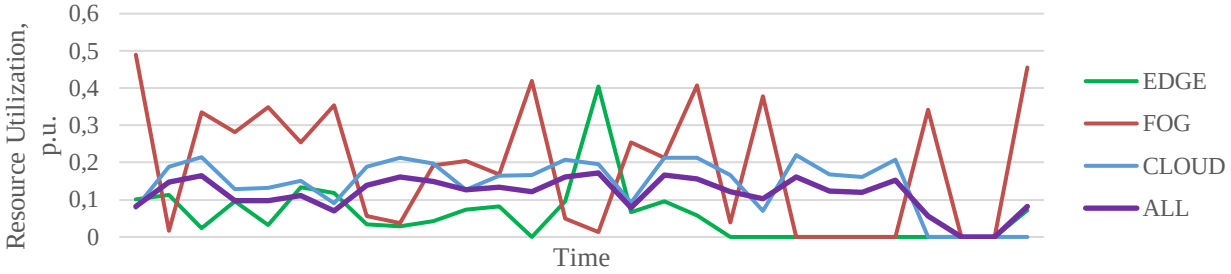


Fig. 9. Resource utilization at each tier and in the system overall in Basic scenario for 50 tasks

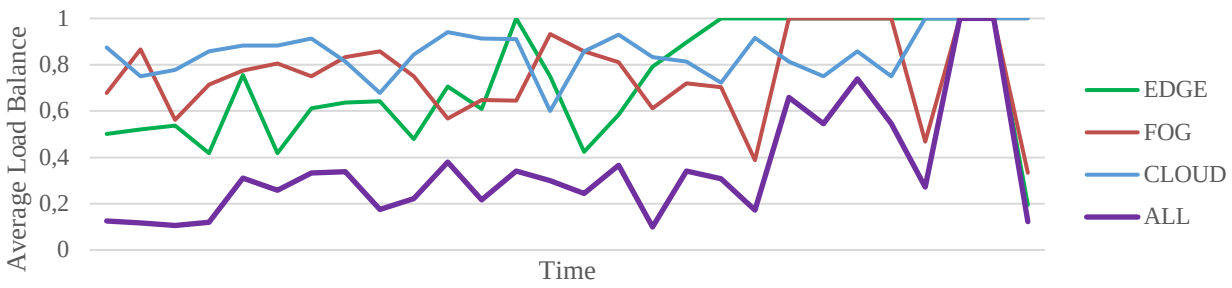


Fig. 10. Average load balancing index for each tier and system overall in Basic scenario for 50 tasks

The highest load was observed at the Fog tier (peak value 49%). For Edge and Cloud tiers, the load did not exceed 22% on average, except for a single spike to 40% for the Edge tier. The overall system load fluctuated between 6% and 17%. These metrics indicate a low system load, with average task characteristics and moderate task generation intervals. The load balancing index has the best values for the Cloud tier (from 0.6 to 0.94) and the worst for the Edge tier (from 0.4 to 0.77 on average without considering individual spikes). This is primarily related to the number of computing nodes at the tier and their processor power. The Edge tier has 6 computing nodes and executes simple tasks, so in this scenario, only part of the nodes is actually needed to process all such tasks, while other nodes remained idle. Meanwhile, the Cloud tier has only two computing nodes that were constantly loaded with tasks. It should be noted that the load balancing index has a value of 1 in the case of perfect balance or when no load is present. This is confirmed by Figures 9 and 10, which show that at the end of the simulation, the tier load decreased to 0, and at the same time, the tier load balancing index acquired a value of 1.

### 5.3 Basic scenario with scaling

In this scenario, the impact of increasing the number of tasks on the efficiency of the allocation method was investigated. Three simulation runs of the basic scenario were performed: 50, 250, and 500 tasks at each tier with unchanged generation intervals. The relationship between the number of tasks initiated at a tier and the number of tasks executed at that tier for all simulation runs is presented in Figure 11.

The proposed method demonstrates stable task allocation regardless of the number of tasks. For the Cloud tier, the share of locally executed tasks is 66%, 65.6%, and 66.2% for scenarios with 50, 250, and 500 tasks, respectively. For the Fog tier, this share is 66% for 50 and 250 tasks, and 63.4% for 500 tasks. For the Edge tier, 100% of tasks are executed locally regardless of quantity.

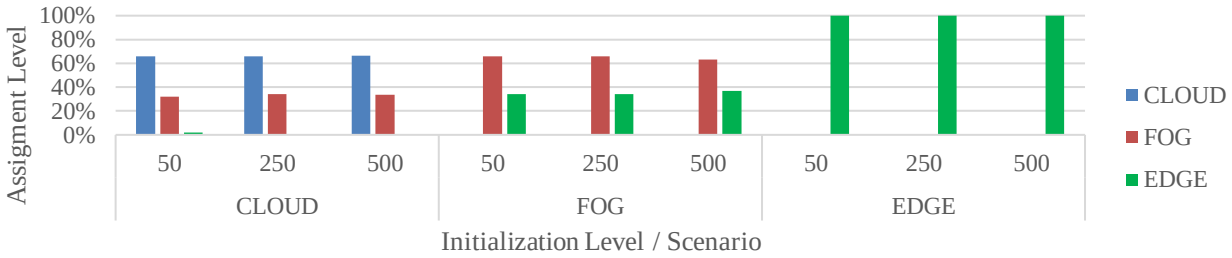


Fig. 11. Ratio of tasks initiated at tier to tasks executed at tier in Basic scenario for 50/250/500 tasks

Average latency values for different numbers of tasks are shown in Figure 12. Assignment latency remains stable at 260 ms for all scenarios. Execution latency shows slight growth from 250 ms for 50 tasks to 270 ms for 500 tasks. Total latency increases from 650 ms to 670 ms, representing a 3% increase with a 10-fold load increase.

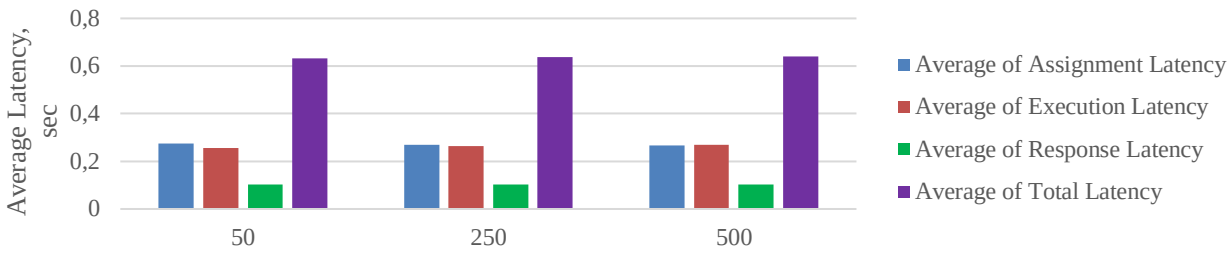


Fig. 12. Average task latency in Basic scenario for 50/250/500 tasks

The dynamics of computing node load and IoT system load balancing index for different numbers of tasks are illustrated in Figures 13 and 14.

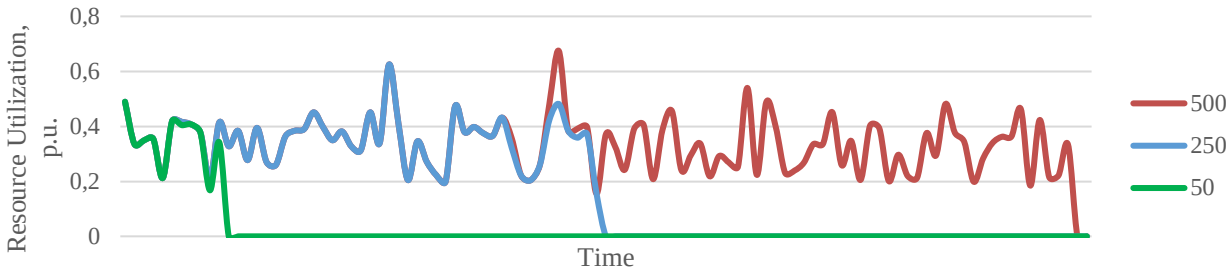


Fig. 13. Resource utilization in Basic scenario for 50/250/500 tasks

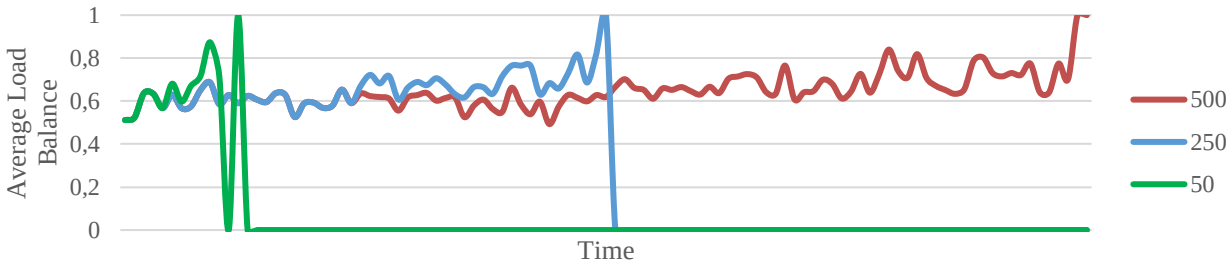


Fig. 14. Average load balancing index in Basic scenario for 50/250/500 tasks

With 50 tasks, peak load does not exceed 49%, with 250 tasks it reaches 62%, and with 500 tasks – 67%. The overall trend demonstrates proportional modest growth without sharp spikes or system overloads. A similar dynamic is observed for the load balancing index, which increases with the number of tasks, indicating better system balance due to greater resource utilization and less computing node idle time.

### 5.4 High scenario with scaling

In the High scenario, task generation intervals were reduced to 100 ms, 200 ms, and 200 ms for Edge, Fog, and Cloud tiers, respectively, resulting in a significantly higher task generation frequency. Simulation was conducted for 50, 250, and 500 tasks at each tier.

The task allocation for this scenario is illustrated in Figure 15, showing that with high generation intensity, task allocation remains similar to the basic scenario, with minor differences. The share of locally executed tasks for the Cloud tier increased to 78%, 77.2%, and 76.4% for 50, 250, and 500 tasks, respectively, while for Edge and Fog tiers, virtually no changes occurred.

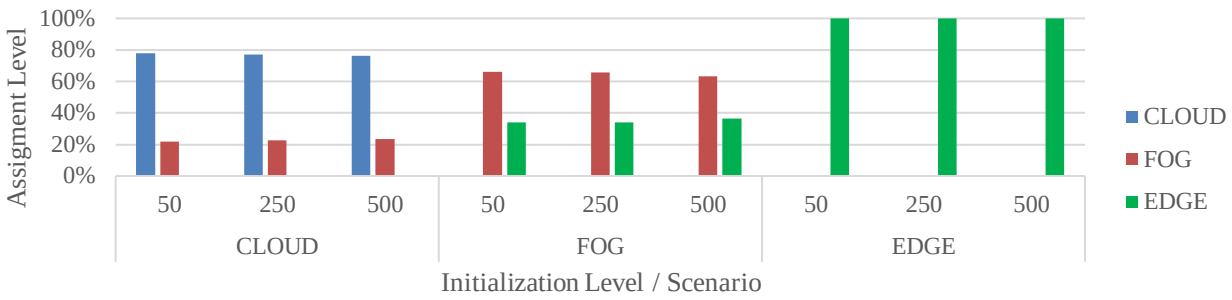


Fig. 15. Ratio of tasks initiated at tier to tasks executed at tier in High scenario for 50/250/500 tasks

The average latency values for different numbers of tasks for this scenario are presented in Figure 16. Task assignment and response latencies remained at the same level as in the basic scenario for 50, 250, and 500 tasks. Task computational latency increased by an average of 0.1s for all three runs compared to the basic scenario. Total task latency increased by an average of 15%.

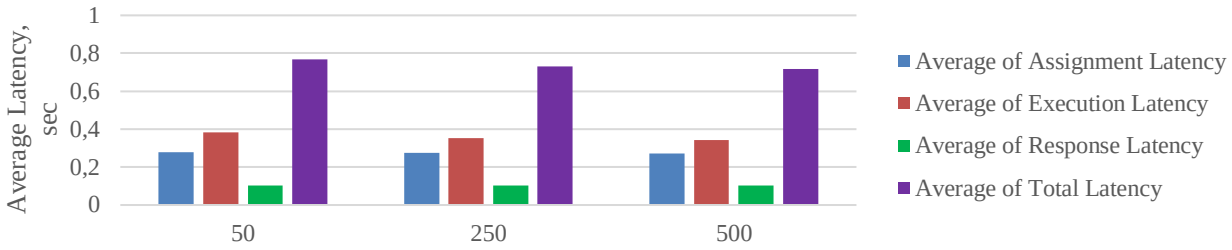


Fig. 16. Average task latency in High scenario for 50/250/500 tasks

The dynamics of computing node load and IoT system load balancing index for different numbers of tasks are shown in Figures 17 and 18.

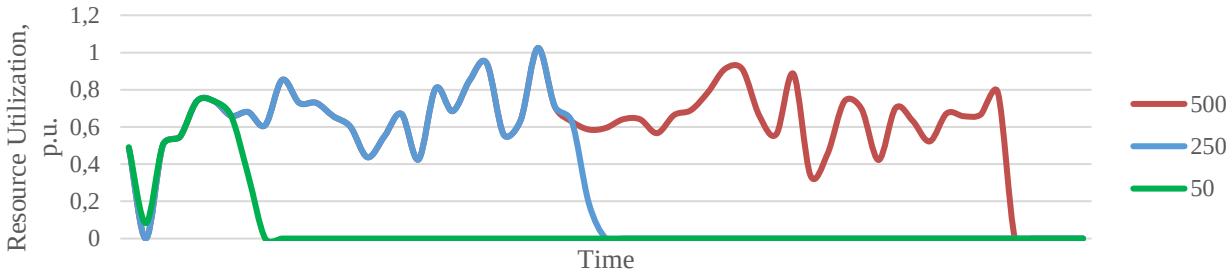


Fig. 17. Resource utilization in High scenario for 50/250/500 tasks

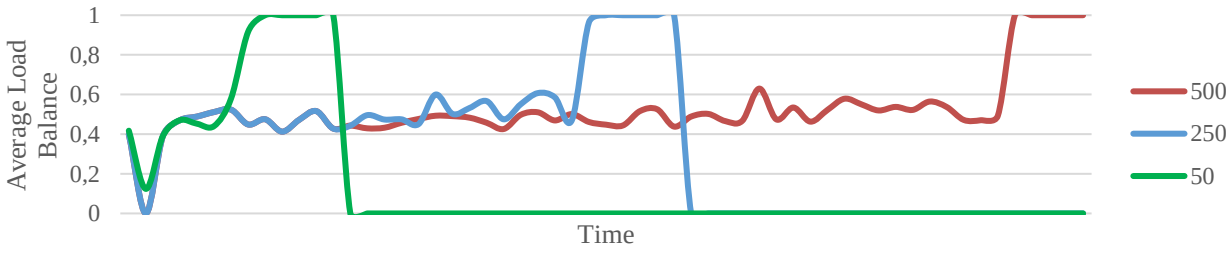


Fig. 18. Average load balancing index in High scenario for 50/250/500 tasks

As demonstrated in Figure 17, more intensive resource utilization in this scenario: peak load with 50 tasks is 73%, with 250 tasks – 102%, and with 500 – 91%. Overall, load fluctuates between 40% and 95%, while the average load balancing index is between 0.4 and 0.6, which may indicate overloading of individual nodes or tiers.

### 5.5 Low scenario with scaling

In the Low scenario, task generation intervals were increased to 800 ms, 1200 ms, and 1500 ms for Edge, Fog, and Cloud tiers, respectively, creating a more sparse task flow. Simulation was conducted for 50, 250, and 500 tasks at each tier.

The ratio between the number of tasks initiated at a tier and the number of tasks executed at that tier for this scenario is presented in Figure 19. The overall trend remains similar to the basic scenario with the only difference being that the share of locally executed tasks for the Cloud tier decreased to 62%, 64.8%, and 65.4% for 50, 250, and 500 tasks, respectively, compared to the basic scenario, indicating lower Fog tier load, allowing it to process part of the tasks. This is confirmed by the percentage of Cloud tasks assigned to the Fog tier.

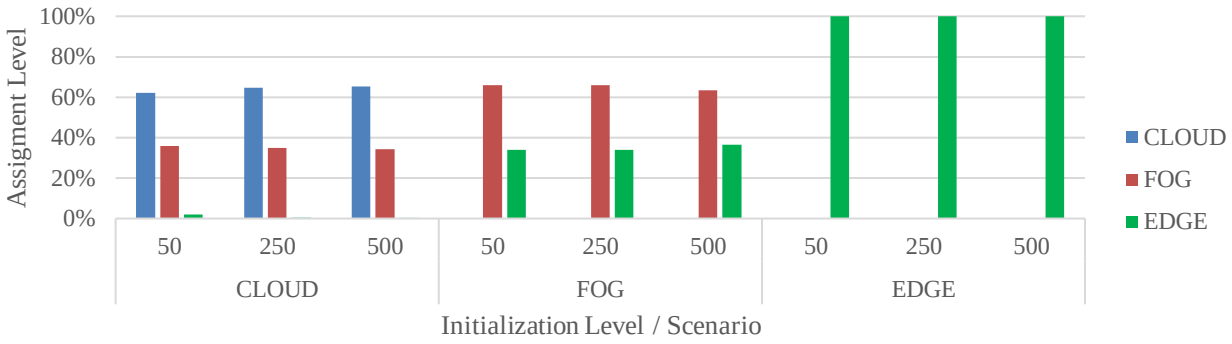


Fig. 19. Ratio of tasks initiated at tier to tasks executed at tier in Low scenario for 50/250/500 tasks

Average latency values for different numbers of tasks for this scenario are shown in Figure 20. Latency values remain close to the basic scenario, with total execution latency for 250 and 500 tasks decreased by 12 ms and 13 ms, respectively. This is related to fewer task routing since resource availability at target task execution tiers increased.

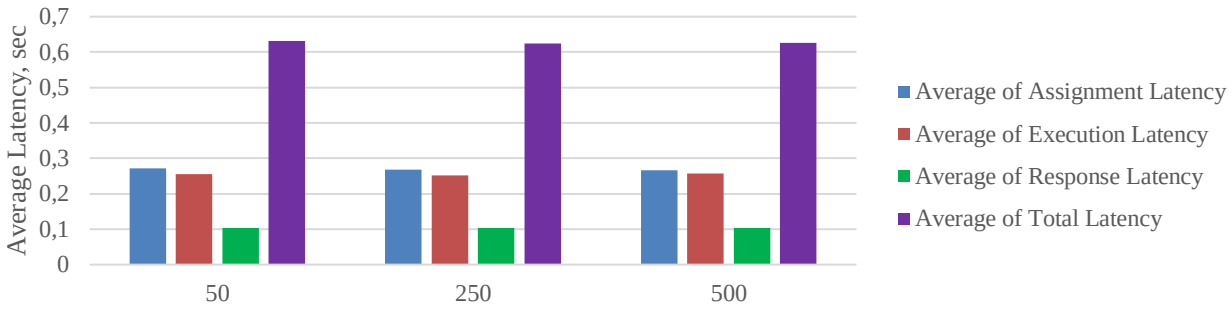


Fig. 20. Average task latency in Low scenario for 50/250/500 tasks

The dynamics of computing node load and IoT system load balancing index for different numbers of tasks are illustrated in Figures 21 and 22.

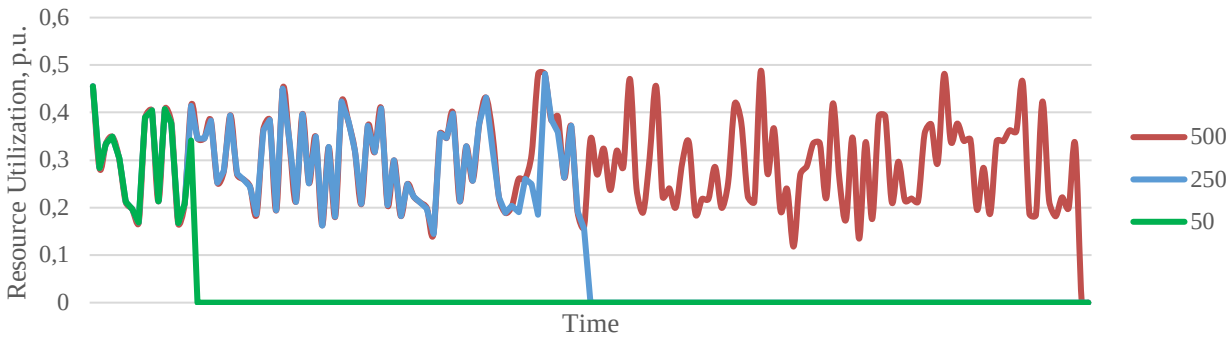


Fig. 21. Resource utilization in Low scenario for 50/250/500 tasks

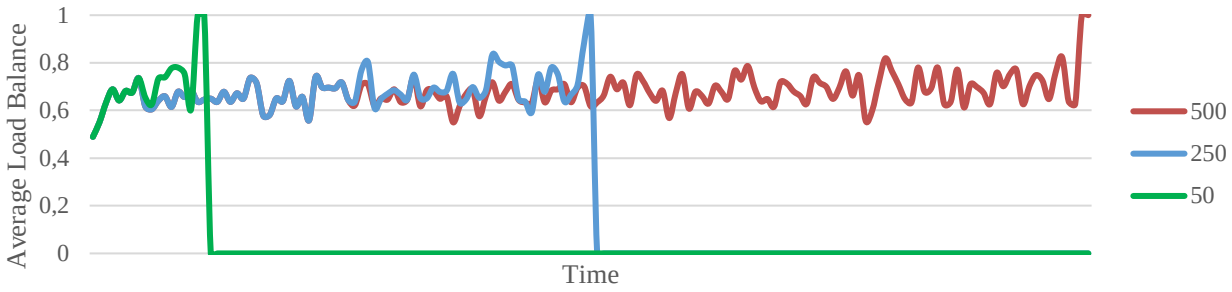


Fig. 22. Average load balancing index in Low scenario for 50/250/500 tasks

As demonstrated in Figure 21, the most uniform resource utilization without sharp peaks compared to other scenarios. The maximum load does not exceed 50% even for 500 tasks, and overall dynamics are characterized by smooth changes within a 20–40% system load range. Average load balancing index values (Fig. 22) are predominantly in the 0.6–8.0 range, demonstrating the best indicators among all scenarios.

### 5.6. Comparative analysis at different task generation intensities

For a detailed analysis of the impact of task generation intensity on system performance, a comparison of three scenarios with the same number of tasks (50 at each tier) but different generation intervals was conducted. Task allocation at different task generation intensities is shown in Figure 23, demonstrating an increase in the share of locally executed tasks for the Cloud tier with decreasing task generation intervals: from 62% for the Low scenario to 78% for the High scenario.

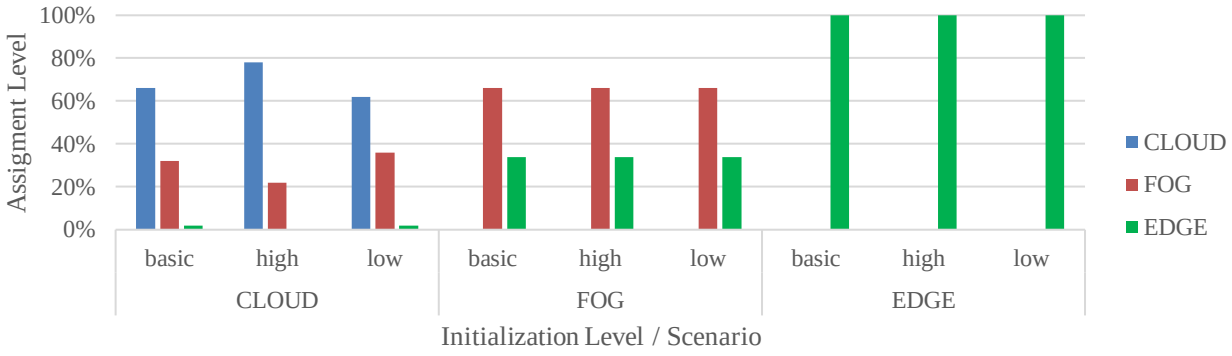


Fig. 23. Ratio of tasks initiated at tier to tasks executed at tier in Basic/High/Low scenarios for 50 tasks

The latency comparison between scenarios is presented in Figure 24. Assignment latency is within 272–279 ms for all scenarios. Response latency also has a virtually identical value for all scenarios. The largest differences are observed in computation latency, which increases by approximately 120 ms with increased task generation intensity due to increased load on computing nodes.

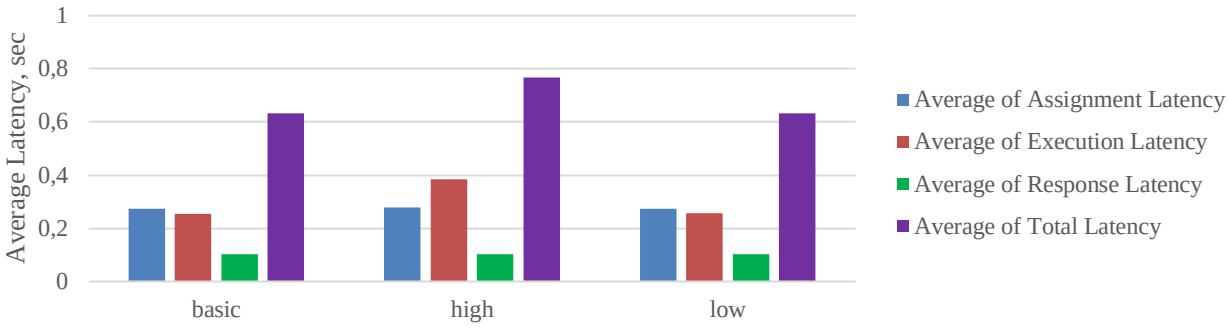


Fig. 24. Average task latency in Basic/High/Low scenarios for 50 tasks

The dynamics of computing node load and IoT system load balancing index for different task generation intensities are shown in Figures 25 and 26.

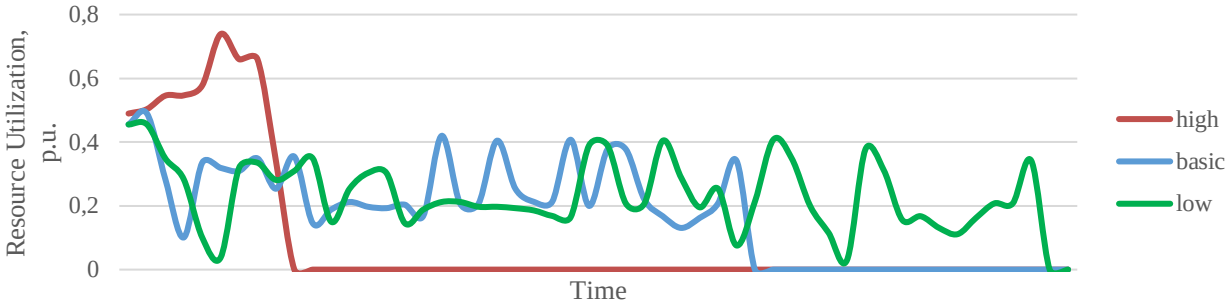


Fig. 25. Resource utilization in Basic/High/Low scenarios for 50 tasks

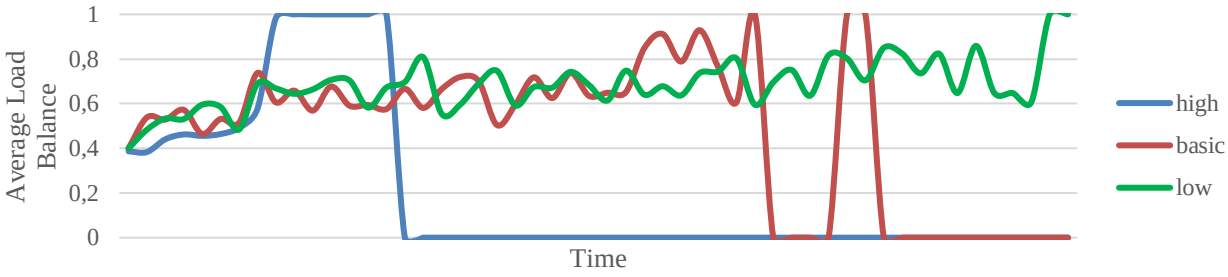


Fig. 26. Average load balancing index in Basic/High/Low scenarios for 50 tasks

The typical dynamics of resource usage for different scenarios are shown in Figure 25. The High scenario is characterized by high short-term resource utilization with a peak value of 73%. Basic and Low scenarios demonstrate moderate fluctuations, averaging between 20% and 40%. The duration of active resource utilization periods also correlates with task generation intervals. The load balancing index (Fig. 26) for the Low scenario shows the most stable and high values (0.6–0.85).

### 5.7 Comparative analysis with different task characteristics

To investigate the adaptability of the method to different task types, three specialized scenarios were conducted with 100 tasks at each tier, but with different emphases on task characteristics. Task allocation for these scenarios is presented in Figure 27, demonstrating expected changes in task allocation when corresponding task parameters change. Thus, with increased computational complexity of tasks, most tasks remain at Cloud and Fog tiers without being redirected to lower hierarchy tiers. With increased latency sensitivity, the majority of tasks (94%) are redirected to the Edge tier. And with high bandwidth requirements, characterized by task data size, tasks were predominantly executed at Fog and Edge tiers.

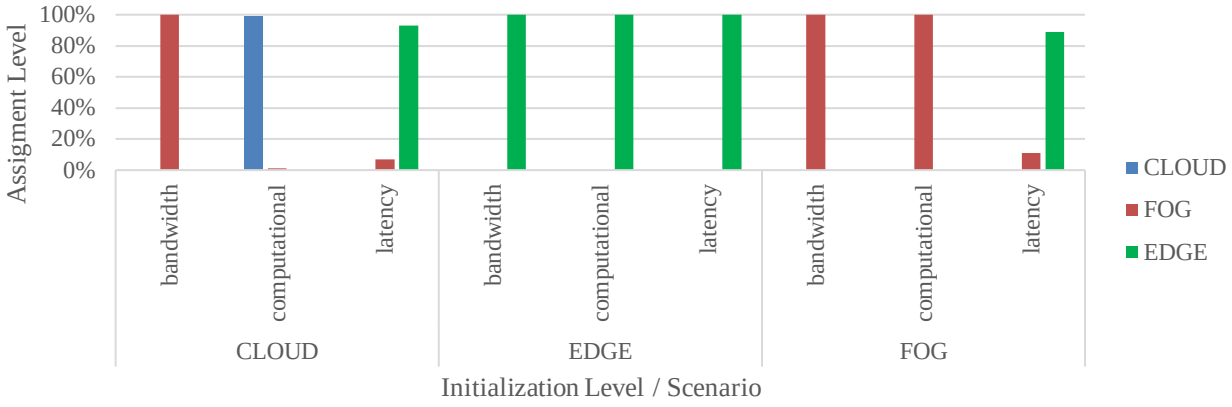


Fig. 27. Ratio of tasks initiated at tier to tasks executed at tier in Compute-Intensive/Latency-Intensive/Bandwidth-Intensive scenarios for 100 tasks

The dynamics of computing node load for different task characteristics are shown in Figure 28.

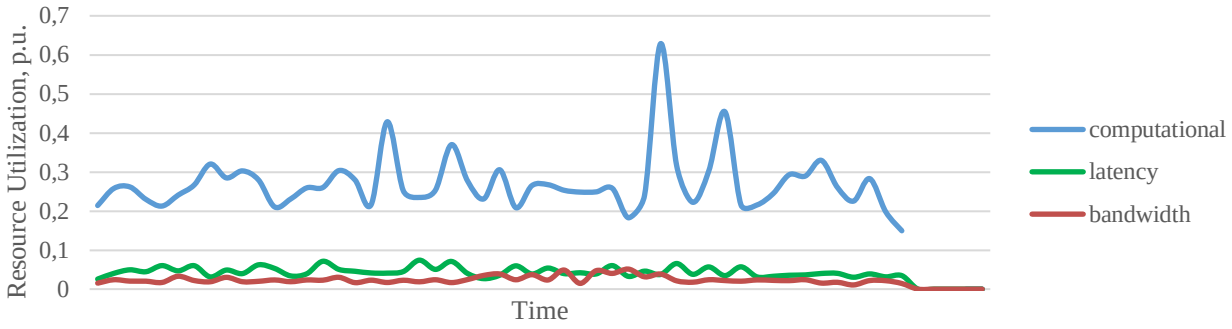


Fig. 28. Resource utilization in Compute-Intensive/Latency-Intensive/Bandwidth-Intensive scenarios for 100 tasks

Tasks with higher computational complexity expectedly lead to increased system load, while latency sensitivity and bandwidth requirements virtually do not affect system load.

## 6. Discussion of obtained results of proposed decentralized task allocation method

Experimental investigation of the developed method using the iFogSim2 simulator has been demonstrated that with a load of 50 tasks per tier at moderate generation intervals, the method ensures predominantly local task execution with low resource utilization up to 20%. When scaling the load to 250 and 500 tasks, the proportion of locally executed tasks remains virtually unchanged, while total task latency increases by 3%, which indicates the stability of the method regardless of the number of tasks. Changes in task generation intensity have shown that at small intervals, the

proportion of local execution for the Cloud tier increases to 76–78%, while at low intensity, optimal utilization of available resources is achieved with improved balance indicators of 0.6–0.8. Task assignment latency and response latency remain virtually unchanged. Total task latency increases by approximately 10–20% when task generation intensity increases due to increased system load. The method demonstrates expected distribution adaptivity for different task types. Computationally intensive tasks are mainly executed on more powerful Cloud and Fog tiers. Latency-sensitive tasks are redirected to the Edge tier in 94% of cases to minimize latency. Tasks with high bandwidth requirements are optimally distributed between Fog and Edge tiers. However, the total task execution latency increases on average by 400 ms when transferring tasks to lower tiers. Additionally, there are extra transport latencies when transferring tasks to another tier and subsequently returning, in cases where the necessary computing power is insufficient. Therefore, further research will be directed toward optimizing the developed task allocation method, specifically inter-tier task routing, to reduce total task execution time by decreasing the number of task redirections between tiers. Additional research is also needed to verify the feasibility of implementing resource release mechanisms during task allocation. Furthermore, a future research direction is the development of MEL management methods for efficient resource utilization, load balancing, and reducing task execution time.

## Conclusions

In this paper, a general model for task allocation and resource management in IoT systems based on osmotic computing has been developed. The considered hierarchical three-tier architecture of distributed IoT systems has been designed to ensure the rational utilization of computing resources at the edge, fog, and cloud tiers, regulate load distribution among them, and improve system adaptivity to dynamic environmental changes.

A decentralized task allocation method using fuzzy logic apparatus has been proposed, which considers both the characteristics and requirements of tasks and the state of computing resources to determine the optimal execution environment. The allocation method includes two decision-making stages using the Mamdani fuzzy inference system: determining the task allocation direction and selecting the optimal computing node for its execution.

Experimental investigation of the developed method were conducted using the iFogSim2 simulator. The results of the investigation showed that the percentage of locally executed tasks remains virtually unchanged with different numbers of tasks, indicating stability in decision-making. An increase in the intensity of task generation leads to an increase in task execution latency due to an increase in the load on the computing nodes, while task assignment latency and task response latency remain unchanged. The method demonstrated the adaptability of distribution for different types of tasks.

## References

[1] C. Cole, “IoT 2024 in review: The 10 most relevant IoT developments of the year”, IoT Analytics, Jan. 15, 2025. [Online]. Available: https://iot-analytics.com/iot-2024-review (Accessed: Apr. 27, 2025).

[2] I. Alfonso, M. Alférez, V. Amaral, and D. Díaz, “Self-adaptive architectures in IoT systems: a systematic literature review”, Journal of Internet Services and Applications, vol. 12, 2021, pp. 1–28, https://doi.org/10.1186/s13174-021-00145-8.

[3] O. Rolik, S. Telenyk, and E. Zharikov, “IoT and Cloud Computing: The Architecture of Microcloud-Based IoT Infrastructure Management System”, in Securing the Internet of Things: Concepts, Methodologies, Tools, and Applications, Hershey, PA, USA: IGI Global, 2020, pp. 1157–1185, https://doi.org/10.4018/978-1-5225-9866-4.ch052.

[4] R. Villari, A. Puliafito, M. Fazio, and M. Paone, “Osmotic Computing: A New Paradigm for Edge/Cloud Integration”, IEEE Cloud Computing, vol. 3, no. 7, 2016, pp. 76–83, https://doi.org/10.1109/MCC.2016.124.

[5] B. Neha, M. Shyamala, K. P. Rajan, M. Krishnaveni, and R. Gnanamurthy, “A Systematic Review

on Osmotic Computing", ACM Transactions on Internet of Things, vol. 3, no. 2, 2022, pp. 1–30, https://doi.org/10.1145/3488247.

[6] A. Mahapatra, K. Mishra, R. Pradhan, and S. Majhi, "Next Generation Task Offloading Techniques in Evolving Computing Paradigms: Comparative Analysis, Current Challenges, and Future Research Perspectives", Archives of Computational Methods in Engineering, 2023, https://doi.org/10.1007/s11831-023-10021-2.

[7] A. Mahapatra, K. Mishra, S. K. Majhi, and R. Pradhan, "Latency-aware Internet of Things Scheduling in Heterogeneous Fog-Cloud Paradigm", in Proceedings of the 3rd International Conference on Emerging Technology (INCET), Belgaum, India, May 27–29, 2022, IEEE, pp. 1–7, https://doi.org/10.1109/INCET54531.2022.9824613.

[8] S. Wang, T. Zhao, and S. Pang, "Task scheduling algorithm based on improved firework algorithm in fog computing", IEEE Access, vol. 8, 2020, pp. 32385–32394, https://doi.org/10.1109/ACCESS.2020.2973758.

[9] T. Zheng, J. Wan, J. Zhang, and C. Jiang, "Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing", Journal of Cloud Computing: Advances, Systems and Applications, vol. 11, article 3, 2022, https://doi.org/10.1186/s13677-021-00276-0.

[10] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing", IEEE Access, vol. 7, 2019, pp. 42735–42744, https://doi.org/10.1109/ACCESS.2019.2907615.

[11] E. H. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller", International Journal of Man-Machine Studies, vol. 7, no. 1, 1975, pp. 1–13, https://doi.org/10.1016/S0020-7373(75)80002-2.

[12] G. J. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic: Theory and Applications. Englewood Cliffs, NJ, USA: Prentice Hall, 1995.

[13] H.J. Zimmermann, Fuzzy Set Theory – and Its Applications, 4th ed. Dordrecht, Netherlands: Springer, 2001, https://doi.org/10.1007/978-94-010-0646-0.

[14] A. Mahapatra, S. K. Majhi, K. Mishra, R. Pradhan, D. C. Rao, and S. K. Panda, "An Energy-Aware Task Offloading and Load Balancing for Latency-Sensitive IoT Applications in the Fog-Cloud Continuum", IEEE Access, vol. 12, 2024, pp. 14334–14349, https://doi.org/10.1109/ACCESS.2024.3357122.

[15] C. Chakraborty, K. Mishra, S. K. Majhi, and H. K. Bhuyan, "Intelligent Latency-Aware Tasks Prioritization and Offloading Strategy in Distributed Fog-Cloud of Things", IEEE Transactions on Industrial Informatics, vol. 19, no. 2, pp. 2099–2106, Feb. 2023, https://doi.org/10.1109/TII.2022.3173899.

[16] W. Jin and A. Rezaeipanah, "Dynamic Task Allocation in Fog Computing Using Enhanced Fuzzy Logic Approaches", Scientific Reports, vol. 15, no. 1, art. 18513, 2025, https://doi.org/10.1038/s41598-025-03621-4.

[17] D. H. Abdulazeez and S. K. Askar, "A Novel Offloading Mechanism Leveraging Fuzzy Logic and Deep Reinforcement Learning to Improve IoT Application Performance in a Three-Layer Architecture within the Fog-Cloud Environment", IEEE Access, vol. 12, 2024, https://doi.org/10.1109/ACCESS.2024.3376670.

[18] S. Javanmardi, G. Sakellari, M. Shojafar, and A. M. Caruso, "Why It Does Not Work? Metaheuristic Task Allocation Approaches in Fog-Enabled Internet of Drones", Simulation Modelling Practice and Theory, vol. 133, 2024, art. 102913, https://doi.org/10.1016/j.simpat.2024.102913.

[19] D. Alsadie, "Advancements in heuristic task scheduling for IoT applications in fog-cloud computing: challenges and prospects," PeerJ Computer Science, vol. 10, art. e2128, 2024, https://doi.org/10.7717/peerj-cs.2128.

[20] E. Hamza, M. Bakhouya, and A. Koubâa, "Multi-objective Fuzzy Approach to Scheduling and Offloading Workflow Tasks in Fog-Cloud Computing", Applied Sciences, vol. 13, no. 15, art. 8785, 2023, http://dx.doi.org/10.1016/j.simpat.2022.102687.

[21] M. Villari, A. Puliafito, M. Fazio, S. Dustdar, R. Ranjan, and S. Bonomi, "Software Defined

Membrane: Policy-Driven Edge and Internet of Things Security", IEEE Cloud Computing, vol. 4, no. 4, 2017, pp. 92–99, https://doi.org/10.1109/MCC.2017.3791014.

[22] R. Jain, D. M. Chiu, and W. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems", arXiv preprint, arXiv:cs/9809099, 1998, https://doi.org/10.48550/arXiv.cs/9809099.

[23] M. R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "iFogSim2: An Extended iFogSim Simulator for Mobility, Clustering, and Microservice Management in Edge and Fog Computing Environments", arXiv preprint, arXiv:2109.05636, 2021, https://doi.org/10.48550/arXiv.2109.05636.

[24] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments", Software: Practice and Experience, vol. 47, no. 9, pp. 1275–1296, Jun. 2017, https://doi.org/10.1002/spe.2509.

[25] R. Mahmud and R. Buyya, "Modelling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit", arXiv preprint, arXiv:1812.00994, Dec. 2018, https://doi.org/10.48550/arXiv.1812.00994.

[26] R. Andreoli, J. Zhao, T. Cucinotta, and R. Buyya, "CloudSim 7G: An Integrated Toolkit for Modeling and Simulation of Future Generation Cloud Computing Environments", Software: Practice and Experience, vol. 53, no. 6, pp. 1041–1058, 2025, https://doi.org/10.1002/spe.3413.

[27] M. A. Shahid, M. M. Alam, M. M. Su'ud, and K. Pratap, "A Systematic Parameter Analysis of Cloud Simulation Tools in Cloud Computing Environments", Applied Sciences, vol. 13, no. 15, art. 8785, 2023, https://doi.org/10.3390/app13158785.

УДК 004.75

# МЕТОД ДЕЦЕНТРАЛІЗОВАНОГО РОЗПОДІЛУ ЗАДАЧ В ІЄРАРХІЧНИХ СИСТЕМАХ IoT З ВИКОРИСТАННЯМ НЕЧІТКОЇ ЛОГІКИ

**Oleksandr Rolik**
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
http://orcid.org/0000-0001-8829-4645

**Dmytro Nahaiko**
National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine
https://orcid.org/0009-0003-3611-3605

Використання туманних та крайових обчислень розширяють обчислювальні потужності системи Інтернет речей (IoT) до краю мережі, сприяючи мінімізації затримок під час виконання задач. Осмотичні обчислення доповнюють розподілені обчислення, забезпечуючи безшовну інтеграцію між обчислювальними середовищами завдяки динамічній міграції мікро-елементів між різними рівнями ієрархії відповідно до поточних умов навантаження та доступності ресурсів. Однак, з урахуванням концепції осмотичних обчислень, актуальним завданням залишається управління розподілом задач в умовах невизначеності, динамічності та гетерогенності середовища IoT. Метою дослідження є підвищення ефективності використання ресурсів та розподілу задач в ієрархічних системах IoT на основі осмотичних обчислень в умовах невизначеності на динамічних змін середовища. Об'єктом дослідження є процеси розподілу задач у багаторівневих системах IoT, що включають хмарні, туманні та крайові обчислення. Предметом дослідження є методи та моделі розподілу задач і управління обчислювальними ресурсами в системах IoT з використанням парадигми осмотичних обчислень.

В статті представлено трирівневу ієрархічну модель керування, побудовану на основі хмарного, туманного та крайового середовищ, яка реалізує централізовано-децентралізоване управління. Кожен рівень представлений набором обчислювальних вузлів та системою управління, яка здійснює локальний розподіл задач, моніторинг стану ресурсів та управління мікро-елементами. Система управління нижче розташованого рівня підпорядковується вищій за ієрархією системі управління. Розроблено метод децентралізованого розподілу задач в ієрархічних системах IoT з використанням апарату нечіткої логіки. Метод розподілу включає два етапи прийняття рішень з використанням системи нечіткого логічного висновку: визначення напрямку розподілу задачі та вибір оптимального обчислюваного вузла для її виконання. Визначення напрямку розподілу задачі здійснюється на основі характеристик задачі, а рейтинг придатності обчислювальних вузлів визначається з урахуванням затримки виконання задачі, ефективності використання ресурсів та балансування навантаження. Задача призначається вузлу з максимальним рейтингом. Використання нечіткої логіки забезпечує прийняття раціональних рішень в умовах невизначеності в реальному часі, що є характерним для високо-гетерогенних та динамічних середовищ IoT.

Експериментальне моделювання та дослідження методу було здійснено з використанням середовища симуляції iFogSim. Результати дослідження показали, що відсоток локально виконаних задач залишається фактично незмінним при різній кількості задач, що свідчить про стабільність прийняття рішень. Збільшення інтенсивності генерації задач призводить до зростання затримки обчислення задачі через збільшення навантаження на обчислювальні вузли, при цьому затримка призначення задачі та затримка відповіді залишаються незмінними. Метод продемонстрував адаптивність розподілу при різних типах задач.

**Ключові слова:** Інформаційні системи, Інтернет речей, IoT, хмарні обчислення, туманні обчислення, крайові обчислення, осмотичні обчислення, розподіл задач, нечітка логіка.