UDC: 004.7-048.34(043.2) DOI: 10.18372/2073-4751.86.21285

Shklyar O. I.,
orcid.org/0009-0001-6524-6357,
e-mail: oleh.shklyar@gmail.com,

Alkema V. V.,
orcid.org/0009-0000-0009-8237,
e-mail: vitalii.alkema@gmail.com

ADAPTIVE APPROACH TO LOAD BALANCING WITH QOS GUARANTEES IN AN EDGE-FOG-CLOUD ENVIRONMENT

National University «Kyiv Aviation Institute»

Introduction

The rapid development of the Internet of Things (IoT) has led to new requirements for the organization of computing infrastructures. Modern IoT systems generate massive flows of heterogeneous data – from industrial equipment sensors and smart transport networks to medical devices and video surveillance systems. According to Ericsson forecasts, the number of connected devices will exceed 29 billion by 2030 [1], demanding fundamentally new approaches to computing resource organization and load management.

Centralized cloud data processing, despite significant computing resources, is unacceptable for applications with strict time constraints – autonomous vehicles, real-time industrial monitoring, and telemedicine. Transmitting all traffic to the cloud causes unacceptable delays, excessive network bandwidth consumption, and increased energy consumption [2]. The concepts of Edge [3] and Fog [4] computing have been proposed as intermediate processing tiers that bring computations closer to data sources and allow a portion of tasks to be executed locally.

The three-tier Edge-Fog-Cloud architecture provides flexible capabilities for distributing the load among resource-constrained but spatially close edge nodes, regional fog nodes with higher performance, and powerful cloud data centers. However, effective utilization of such heterogeneous infrastructure requires intelligent load balancing – a mechanism for dynamically distributing tasks among nodes considering their characteristics, current state, and QoS requirements.

Existing approaches to load balancing have significant limitations. Classical algorithms – Round Robin, Min-Min, Max-Min – were developed for homogeneous cloud environments and do not account for the multi-tier specifics of Edge-Fog-Cloud [5]. Heuristic and metaheuristic methods (genetic algorithms, PSO, ACO) provide high-quality solutions in static conditions but entail excessive computational overhead for environments with dynamic loads [6]. Most QoS-oriented approaches consider only one or two architectural tiers and do not adapt to load fluctuations [7, 8].

Literature analysis [7–11] indicates that the key unresolved problem is the lack of adaptive mechanisms capable of simultaneously: (1) making decisions in near real-time; (2) accounting for the heterogeneity of the three-tier infrastructure; (3) guaranteeing the fulfillment of task QoS requirements under variable loads.

The aim of this article is to develop an adaptive approach to load balancing in an Edge-Fog-Cloud environment that ensures the fulfillment of task QoS requirements under dynamic load conditions through the efficient selection of the processing tier and node.

Materials and methods

Mathematical Model of the Edge-Fog-Cloud System. Let us consider a three-tier computing environment, formally described by the tuple:

$$S = \langle E, F, C, T, Q \rangle,$$

where $E = \{e_1, e_2, \dots, e_{N_E}\}$is the set of edge nodes (Edge); $F = \{f_1, f_2, \dots, f_{N_F}\}$ is the set of fog nodes (Fog); $C = \{c_1, c_2, \dots, c_{N_C}\}$ is the set of cloud nodes (Cloud); T is the set of tasks; Q is the set of QoS metrics. The overall set of nodes is denoted as $N = E \cup F \cup C$.

Each node $n_j \in N$ is characterized by a vector of hardware parameters:

$$n_j = \left\langle MIPS_j, M_j, BW_j^{\text{in}}, BW_j^{\text{out}}, P_j^{\text{idle}}, P_j^{\text{peak}}, R_j \right\rangle,$$

where $MIPS_j$ is the computing power (Million Instructions Per Second); M_j is the RAM capacity (MB); $BW_j^{\text{in}}, BW_j^{\text{out}}$ are the input and output network bandwidths (Mbps); P_j^{idle}, P_j^{peak} are the power consumption in idle mode and at peak load (W); $R_j \in [0,1]$ is the node reliability coefficient.

The current state of node n_j at time t is described by the vector:

$$\sigma_j(t) = \langle \rho_j(t), m_j^{free}(t), Q_j^{len}(t) \rangle,$$

where $\rho_j(t) \in [0,1]$ is the CPU utilization ratio; $m_j^{free}(t)$ is the available free memory (MB); $Q_j^{len}(t)$ is the task queue length.

A task $t_i \in T$ is described by the tuple:

$$t_i = \langle r_i, d_i, \delta_i^{\max}, \psi_i \rangle,$$

where r_i is the computational complexity (MI); d_i is the input/output data volume (MB); $\delta_i^{\max}$ is the maximum tolerable execution latency (deadline, ms); $\psi_i \in$ {critical,normal,background} is the task priority class.

System QoS Metrics. The set Q includes the following quality indicators:

- Execution latency $L(t_i, n_j)$ - the total time from task arrival to result reception;
- Throughput $P(n_j)$ - the number of tasks processed by the node per unit of time;
- Energy consumption $E(t_i, n_j)$ - the amount of energy spent on executing the task;
- Failure rate $F_r(n_j)$ - the proportion of tasks not executed due to node failure;
- QoS-compliance F_{QoS} - the proportion of tasks executed while meeting the deadline $\delta_i^{\max}$.

Formalization of the Load Balancing Problem. Latency model. Task t_i execution time directly on noden_j:

$$T_{exec}(t_i, n_j) = \frac{r_i}{MIPS_j \cdot (1 - \rho_j(t))}.$$

Data transfer time between source noden_s and selected noden_j:

$$T_{trans}(t_i, n_s, n_j) = \frac{d_i}{min(BW_s^{\text{out}}, BW_j^{\text{in}})} + L_{\text{prop}}(n_s, n_j)$$

where $L_{\text{prop}}(n_s, n_j)$ is the signal propagation time between nodes (determined by network topology). Total task execution latency:

$$L(t_i, n_j) = T_{trans}(t_i, n_s, n_j) + T_{exec}(t_i, n_j) + T_{trans}(t_i, n_j, n_s).$$

Energy consumption model. Current power consumption of the node at load ρ_j:

$$P_j^{exec}(\rho_j) = P_j^{idle} + \rho_j \cdot (P_j^{peak} - P_j^{idle}).$$

Energy consumption for executing taskt_i on noden_j:

$$E(t_i, n_j) = P_j^{exec}(\rho_j) \cdot T_{exec}(t_i, n_j) + P_j^{trans} \cdot T_{trans}(t_i, n_s, n_j),$$

where P_j^{trans} is the power of the node's network interface.

Statement of the optimization problem. The load balancing problem consists of constructing an assignment function$A: T \to N$, which for each taskt_i determines the optimal node $n_j = A(t_i)$. The objective function (1) minimizes the integral quality indicator:

$$J = \frac{\alpha \cdot \overline{L}}{\overline{L}ref} + \frac{\beta \cdot Etotal}{E_{ref}} - \gamma \cdot \overline{U} + \kappa \cdot (1 - F_{QoS}), (1)$$

where $\overline{L}$ is the average task execution latency; E_{total} is the total energy consumption; $\overline{U} = \frac{1}{N}\sum_{j=1}^{N} \rho_j$ is the average load coefficient; $\overline{L}ref$, $Eref$ are normalization constants; $\alpha, \beta, \gamma, \kappa \geq 0$ are weight coefficients, $\alpha + \beta + \gamma + \kappa = 1$.

Problem constraints are formalized as inequalities (2)-(4). Condition (2) guarantees meeting time deadlines, (3) ensures the computational capacity of the node is not exceeded, and (4) limits the available RAM:

$$L\left(t_i, n_j^*\right) \le \delta_i^{max}, \forall t_i \in T, (2)$$

$$\sum_{t_i \in \text{active}(n_j)} r_i \le \text{MIPS}_j, \forall n_j \in N, (3)$$

$$\sum_{t_i \in \text{active}(n_j)} d_i \le M_j, \forall n_j \in N. (4)$$

Architecture of the adaptive approach. The proposed adaptive approach is implemented through a four-component architecture, as shown in Figure 1.

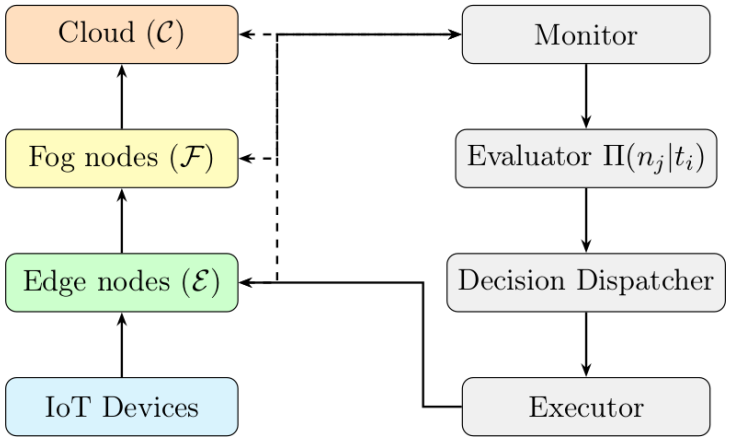

Fig. 1. Architecture of the adaptive load balancing approach

Monitor performs continuous collection of the state vector $\sigma_j(t)$ for each node via distributed agents. Polling frequency depends on the level: Edge - 500 ms, Fog - 1 s, Cloud - 5 s.

Evaluator calculates the priority function of noden_j for taskt_i using Equation (5):

$$\Pi\left(n_j|t_i\right) = w_1 \cdot A_{comp}\left(n_j\right) + w_2 \cdot A_{lat}\left(t_i, n_j\right) + w_3 \cdot A_{en}\left(n_j\right) + w_4 \cdot R_j, (5)$$

where normalized partial indicators:

$$A_{comp}\left(n_j\right) = 1 - \rho_j(t),$$

$$A_{lat}\left(t_i, n_j\right) = max\left(0{,}1 - \frac{L(t_i, n_j)}{\delta_i^{max}}\right),$$

$$A_{en}\left(n_j\right) = 1 - \frac{P_j^{exec}(\rho_j) - P_j^{idle}}{P_j^{peak} - P_j^{idle}},$$

and weights$w_k > 0$ satisfy the condition of summing to 1 and are configured according to the task class ψ_i(Table 1).

Table 1. Configuration of priority function weight coefficients by task classes

Class ψ_i	w_1 (load)	w_2 (latency)	w_3 (energy)	w_4 (reliability)
critical	0.20	0.50	0.10	0.20
normal	0.30	0.35	0.20	0.15
background	0.25	0.15	0.40	0.20

Decision Dispatcher applies a hierarchical strategy for node selection with thresholds Θ_{edge} and Θ_{fog}. Executor transfers the task to the selected node, monitors execution progress, and initiates migration in case of QoS violation.

Algorithm of adaptive load balancing (AABL)

Input: Taskt_i; setsE, F, C; weights $\{w_k\}$; thresholds $\Theta_{edge}, \Theta_{fog}$; adaptation coefficient η.

Output: Assignment $A(t_i) \in N$.

Step 1. Node state collection:

for all$n_j \in N$do

Obtain $\sigma_j(t) = <\rho_j(t), m_j^{free}(t), Q_j^{len}(t)>$

end for

Step 2. Selection of feasible nodes:

$N_{\text{feas}} \leftarrow \{n_j \in N | m_j^{\text{free}}(t) \geq d_i \text{and} L(t_i, n_j) \leq \delta_i^{\max}\}$

if $N_{feas} = \emptyset$ then

return $A(t_i) \leftarrow \arg\min_{n_j \in C} L(t_i, n_j)$

end if

Step 3. Calculation of priority scores:

for all $n_j \in N_{feas}$ do

Calculate $\Pi(n_j|t_i)$ according to formula (5)

end for

Step 4. Hierarchical node selection:

$n_E^* \leftarrow \arg\max_{n_j \in N_{\text{feas}} \cap E} \Pi(n_j|t_i)$

if $\Pi(n_E^*|t_i) \geq \Theta_{edge}$then

return $A(t_i) \leftarrow n_E^*$

end if

$n_F^* \leftarrow argmax_{n_j \in N_{feas} \cap F} \Pi(n_j|t_i)$

if $\Pi(n_F^*|t_i) \geq \Theta_{fog}$ then

return $A(t_i) \leftarrow n_F^*$

end if

return $A(t_i) = argmax_{n_j \in N_{feas} \cap C} \Pi(n_j|t_i)$

Step 5. Thresholds adaptation:

for all $level \in \{edge, fog\}$do

$\Theta_{level}(t+1) \leftarrow \Theta_{level}(t) + \eta \cdot \left(\overline{\rho_{level}}(t) - \overline{\rho}_{level}^{ref}\right)$

end for

Algorithm complexity. The step of selecting suitable nodes and calculating priority scores is executed in $O(N)$. The overall complexity for processing a batch of K tasks is $O(K \cdot N)$, which is acceptable for real-time tasks.

Adaptive thresholds mechanism. Upon node overload $(\rho_j > \rho_{max} = 0.85)$, the Dispatcher initiates migration of tasks with the lowest Π score to nodes of the next level. The parameter $\eta = 0.05$ ensures a smooth system response without oscillations.

Results

Simulation was conducted in the iFogSim environment [9] - an extension of CloudSim [12] for simulating IoT and Fog environments. The setup configuration is given in Table 2.

Table 2. Simulation setup parameters

Parameter	Edge	Fog	Cloud
Number of nodes	20	5	2
Computing power (MIPS)	500–1000	5 000–10 000	50 000–100 000
Memory (MB)	512–2 048	4 096–16 384	65 536–262 144
Bandwidth (Mbps)	100	1 000	10 000
Transmission latency (ms)	2–5	10–20	50–100
P^{idle}/P^{peak} (W)	10 / 30	40 / 120	150 / 500

The task flow was generated according to a Poisson distribution for three scenarios: low load (50 tasks/min), average (150 tasks/min), and peak (300 tasks/min). Each scenario lasted 60 min. Initial AABL thresholds: $\Theta_{edge} = 0.60$, $\Theta_{fog} = 0.45$. For comparison, Round Robin (RR), Random, and Min-Min algorithms were used.

The comparison results in terms of latency, makespan, and energy consumption are presented in Table 3.

The dynamics of average latency depending on the load intensity are shown in Figure 2.

Table 3. Comparison of algorithms by latency and energy consumption

Scenario	Algorithm	Avg. latency (ms)	Makespan (s)	Energy (kJ)
Low load	Round Robin	185	84	42,3
	Random	213	96	48,1
	Min-Min	162	73	37,8
	AABL	118	54	29,4
Avg load	Round Robin	310	142	89,6
	Random	381	175	105,4
	Min-Min	265	121	78,2
	AABL	183	84	58,1
Peak load	Round Robin	518	237	156,2
	Random	647	296	194,7
	Min-Min	441	202	135,1
	AABL	291	133	98,3

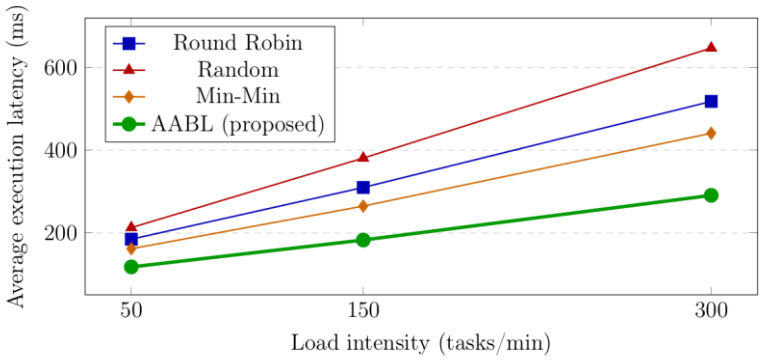

Fig. 2. Average execution latency versus load intensity

The comparison results in terms of QoS-compliance, average utilization ratio, and the number of deadline misses (SLA violations) are presented in Table 4. Figure 3 illustrates the proportion of successfully executed tasks (QoS-compliance) depending on the algorithm and load scenario. An analysis of task distribution across the tiers (Table 5) indicates that AABL effectively utilizes Edge resources for tasks with strict deadlines, thereby reducing the load on the Cloud.

Table 4. Comparison by QoS-compliance and resource utilization

Scenario	Algorithm	QoS-compliance (%)	Avg. load (%)	SLA violations
Low load	Round Robin	91,2	62,4	44
	Random	86,7	57,9	67
	Min-Min	94,1	68,3	29
	AABL	98,4	81,2	8
Avg load	Round Robin	78,9	70,8	211
	Random	71,5	65,1	284
	Min-Min	83,7	75,4	164
	AABL	95,2	88,6	48
Peak load	Round Robin	62,3	77,5	375
	Random	53,8	69,7	461
	Min-Min	70,9	82,1	291
	AABL	89,1	91,8	109

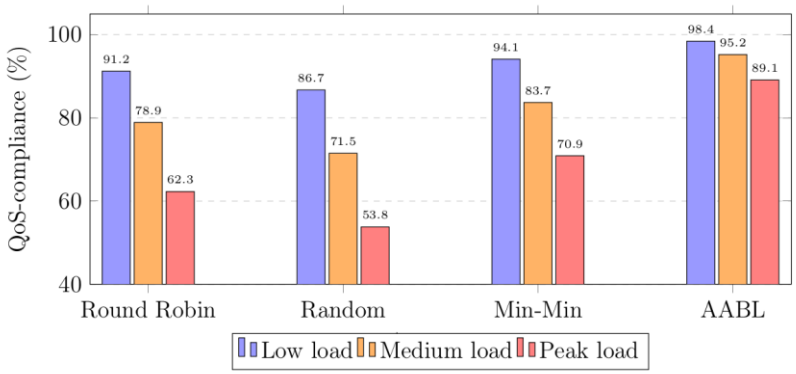

Fig. 3. QoS compliance rate for different algorithms and load scenarios

Table 5. Distribution of tasks among levels under average load (%)

Algorithm	Edge (%)	Fog (%)	Cloud (%)
Round Robin	33,3	33,3	33,3
Random	34,1	32,7	33,2
Min-Min	28,6	35,4	36,0
AABL	51,3	30,2	18,5

Conclusions

The article proposes an adaptive load balancing algorithm (AABL) for a three-tier Edge-Fog-Cloud environment, differing from existing methods by comprehensive consideration of QoS metrics during node selection and dynamic parameter self-tuning.

The obtained results confirm the effectiveness of the proposed approach across all analyzed metrics.

The reduction in latency by 36-44% compared to Round Robin is explained by three factors. First, the priority evaluation function considers the current load state of nodes and avoids placing tasks on overloaded nodes. Second, the hierarchical strategy favors the Edge level for tasks with strict deadlines, reducing data transfer time. Third, the adaptive thresholds mechanism allows the system to react to load increases in advance.

The reduction in energy consumption by 30-37% is achieved by more uniform node loading and a smaller volume of cross-tier data transfer due to local processing on Edge. The increase in the average load coefficient to 81-92% indicates more efficient resource utilization without exceeding saturation thresholds.

The significant reduction in the number of SLA violations is explained by the fact that the deadline condition is an explicit constraint in the optimization problem and is checked at the stage of selecting suitable nodes.

Approach limitations. Monitoring agents introduce additional network load (estimated at 2-4%). Weight coefficients and initial thresholds require tuning for specific applications. The algorithm does not predict future load, limiting its effectiveness during sharp non-stationary disturbances.

Practical application areas: industrial IIoT systems; smart cities with geographically distributed infrastructure; telemedicine systems.

Main research results:

1. A mathematical model of the system has been developed, covering the characteristics of nodes across all three levels, task flow parameters, and QoS constraints.

2. A node priority evaluation function is proposed, integrating computing load, network latency, energy

efficiency, and reliability with weights adapted to the task class.

3. The AABL algorithm provides a 36-44% reduction in average latency and a 30-37% reduction in energy consumption compared to Round Robin; the QoS-compliance ratio is 89-98%.

4. The adaptive thresholds mechanism ensures stable system operation under peak loads.

References

1. Ericsson Mobility Report, November 2025. URL: https://www.ericsson.com/4aca6f/assets/local/reports-papers/mobility-report/documents/2025/ericsson-mobility-report-november-2025.pdf (дата звернення: 04.05.2026).

2. Buyya R., Yeo C. S., Venugopal S., Broberg J., Brandic I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems. 2009. Vol. 25, No. 6. P. 599–616. DOI: 10.1016/j.future.2008.12.001.

3. Shi W., Cao J., Zhang Q., Li Y., Xu L. Edge Computing: Vision and Challenges. IEEE Internet of Things Journal. 2016. Vol. 3, No. 5. P. 637–646. DOI: 10.1109/JIOT.2016.2579198.

4. Bonomi F., Milito R., Zhu J., Addepalli S. Fog computing and its role in the Internet of Things. Proceedings of the 1st Edition of the MCC Workshop on Mobile Cloud Computing (MCC'12), Helsinki, 2012. P. 13–16. DOI: 10.1145/2342509.2342513.

5. Adhikari M., Amgoth T., Srirama S. N. A survey on scheduling strategies for workflows in cloud environment and emerging trends. ACM Computing Surveys. 2019. Vol. 52, No. 4. Art. 69. DOI: 10.1145/3325097.

6. Zhu Z., Zhang G., Li M., Liu X. Evolutionary Multi-Objective Workflow Scheduling in Cloud. IEEE Transactions on Parallel and Distributed Systems. 2016. Vol. 27, No. 5. P. 1344–1357. DOI: 10.1109/TPDS.2015.2446459.

7. Mahmud R., Kotagiri R., Buyya R. Fog computing: A taxonomy, survey and future directions. Internet of everything / ed. by B. Di Martino. Singapore: Springer, 2018. P. 103–130. DOI: 10.1007/978-981-10-5861-5_5.

8. Yousefpour A., Fung C., Nguyen T. et al. All One Needs to Know about Fog Computing and Related Edge Computing Paradigms. Journal of Systems Architecture. 2019. Vol. 98. P. 289–330. DOI: 10.1016/j.sysarc.2019.02.009.

9. Gupta H., Dastjerdi A. V., Ghosh S. K., Buyya R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Software: Practice and Experience. 2017. Vol. 47, No. 9. P. 1275–1296. DOI: 10.1002/spe.2509.

10. Naha R. K., Garg S., Georgakopoulos D. et al. Fog Computing: Survey of Taxonomies, Scenarios, and Future Directions. IEEE Access. 2018. Vol. 6. P. 76543–76564. DOI: 10.1109/ACCESS.2018.2877850.

11. Dastjerdi A. V., Buyya R. Fog computing: Helping the Internet of Things realize its potential. IEEE Computer. 2016. Vol. 49, No. 8. P. 112–116. DOI: 10.1109/MC.2016.245.

12. Calheiros R. N., Ranjan R., Beloglazov A., De Rose C. A. F., Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software: Practice and Experience. 2011. Vol. 41, No. 1. P. 23–50. DOI: 10.1002/spe.995

Shklyar O. I., Alkema V. V.

ADAPTIVE APPROACH TO LOAD BALANCING WITH QOS GUARANTEES IN AN EDGE-FOG-CLOUD ENVIRONMENT

The article proposes an adaptive approach to load balancing in heterogeneous Edge-Fog-Cloud computing environments ensuring Quality of Service (QoS) guarantees. A mathematical model of a three-tier infrastructure is developed, formalizing the characteristics of computing nodes, task flows, and QoS constraints. A node priority evaluation function is proposed, taking into account computing load, network latency, energy efficiency, and reliability; the weight coefficients are dynamically adapted to the task class. The algorithm makes decisions on task placement across Edge, Fog, and Cloud tiers based on the current system state and task QoS requirements. Simulation results in the iFogSim environment compared to Round Robin, Random, and Min-Min algorithms demonstrate a 36–44% reduction in average execution latency, a 30–37% decrease in energy consumption, and an increase in the QoS compliance ratio to 89–98% under various load scenarios.

Keywords: *load balancing, QoS, edge computing, fog computing, cloud computing, IoT, adaptive algorithm.*

Шкляр О. І., Алькема В. В.

АДАПТИВНИЙ ПІДХІД ДО БАЛАНСУВАННЯ НАВАНТАЖЕННЯ З ГАРАНТІЯМИ QOS У СЕРЕДОВИЩІ EDGE–FOG–CLOUD

У статті запропоновано адаптивний підхід до балансування навантаження в гетерогенних обчислювальних середовищах Edge–Fog–Cloud із забезпеченням гарантій якості обслуговування (QoS). Розроблено математичну модель трирівневої інфраструктури з формалізацією характеристик обчислювальних вузлів, потоків задач та QoS-обмежень. Запропоновано функцію пріоритетного оцінювання вузлів, що враховує обчислювальне навантаження, мережеву затримку, енергоефективність та надійність; вагові коефіцієнти динамічно адаптуються до класу задачі. Алгоритм приймає рішення щодо розміщення задач між рівнями Edge, Fog та Cloud на основі поточного стану системи та QoS-вимог задач. Результати моделювання у середовищі iFogSim у порівнянні з алгоритмами Round Robin, Random та Min-Min демонструють скорочення середньої затримки виконання на 36–44%, зниження енергоспоживання на 30–37% та підвищення коефіцієнта виконання QoS-вимог до 89–98% у різних сценаріях навантаження.

Ключові слова: *балансування навантаження, QoS, граничні обчислення, туманні обчислення, хмарні обчислення, IoT, адаптивний алгоритм.*

Стаття подана до редакції: 14*/05/2026*
Стаття прийнята до опублікування: 18/05/2026
Стаття опублікована: 30/05/2026