Information systems research

UDC 004.7 **doi:** https://doi.org/10.20998/2522-9052.2024.4.07

Nina Kuchuk[1], Svitlana Kashkevich[2], Viacheslav Radchenko[3], Yuliia Andrusenko[3], Heorhii Kuchuk[1]

[1] National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine
[2] National Aviation University, Kyiv, Ukraine
[3] Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

APPLYING EDGE COMPUTING IN THE EXECUTION IoT OPERATIVE TRANSACTIONS

Abstract. Topicality. IoT information processing is usually performed in a cloud environment. However, this creates problems associated with delays in data transfer to the cloud. It is especially important to reduce these delays when processing operational IoT transactions. This can be achieved by transferring part of the calculations to IoT peripheral devices. However, it is necessary to take into account the specific features of embedded IoT systems. **The subject of study** in the article is methods for transferring the load to IoT peripheral devices. The purpose of the article is to reduce the execution time of operational IoT transactions by increasing the efficiency of the system infrastructure by transferring part of the computing load to IoT peripheral devices. **The following results** were obtained. A conclusion has been made about the possibility of constructing a distributed information system based on Internet of Things devices. A model of a computing node has been formed, which made it possible to specify a separate computing node, taking into account its location and functioning features. A method for distributing tasks among the nodes of a distributed information system has been developed. The method allows taking into account the features of each computing node and the state of communication channels between them. The developed algorithm for implementing the method is based on the analysis of a stationary or non-stationary environment and changing the greedy strategy of the agent. **Conclusion.** Studies of the effectiveness of the proposed method have been conducted. The simulation results have shown that the proposed method can significantly reduce the processing time of operational transactions.

Keywords: Internet of Things; Computer System; Fog Layer; edge computing; operative transactions.

Introduction

Today, embedded systems are widely developed and widespread [1]. In particular, Internet of Things (IoT) systems are gaining popularity [2]. The main idea of IoT is to embed computing modules into objects of the surrounding world [3]. Such systems allow collecting information about the state and functioning of a large number of objects. This information is then transmitted over a communication network for further processing and analysis. The Internet of Things has become widespread in various fields [4–6]. IoT information processing is usually performed in a cloud environment.

However, this creates problems associated with delays in data transfer to the cloud. It is especially important to reduce these delays when processing operational IoT transactions. This can be achieved by transferring some of the calculations to IoT peripheral devices [7, 8].

IoT information processing is usually performed in a cloud environment. However, this creates problems associated with delays in data transfer to the cloud. It is especially important to reduce these delays when processing operational IoT transactions. This can be achieved by transferring some of the calculations to IoT peripheral devices [7, 8].

Information from IoT sensors is received by peripheral layer devices. Since these devices are computing nodes, it is possible to build a distributed system with computing nodes based on IoT devices (PC IoT). In addition to the problems inherent in classical distributed computing systems, such a system adds features of embedded systems [9–11]:

– independent power supplies,
– low data rates,
– use of low-power processors,
– high interference levels in communication channels,
– constant movement of devices,
– heterogeneity of computing nodes.

Therefore, the use of task distribution methods used in classical distributed systems is impractical [12]. The solution to this problem can be achieved by using methods that can take into account the features of IoT devices [13].

Literature review. Let's consider some scientific works on this topic.

In the article [14], only homogeneous media are considered.

In articles [15, 16], task distribution occurs only in the cloud.

The method proposed in article [17] is focused only on simple topologies IoT.

The algorithm proposed in the article [18] does not take into account the heterogeneity of the environment. Similar problems arise when applying the algorithm given in the article [19].

The algorithm given in the article [20] is not oriented to the real-time mode. The methods proposed in articles [21, 22] do not take into account the limited resources of the edge computing.

Articles [23, 24] do not take into account the costs associated with data transmission.

Algorithms for IoT flows are proposed in articles [25, 26]. But it is used only in a homogeneous environment. The algorithms proposed in articles [27,

© Kuchuk N., Kashkevich S., Radchenko V., Andrusenko Y., Kuchuk H., 2024

28] are focused only on structures similar to the structures of the cloud environment. The algorithm based on deep learning proposed in article [29] is effective only for a homogeneous environment, as is the algorithm used in article [30]. A strategy based on deep learning with the pooling of resource capabilities is proposed in the article [31]. But it can't be implemented on a edge layer with limited device capabilities.

Consequently, the considered scientific works when implementing the load redistribution process don't sufficiently take into account the features edge computing. Therefore, it is advisable to develop an appropriate method.

The purpose of the study is to reduce the time required for performing operational IoT transactions by increasing the efficiency of the system infrastructure by transferring part of the computing load to the IoT peripherals.

1. Modeling the process of assigning operational transactions to IoT nodes

1.1. Setting the task of assigning transactions to IoT nodes. The assignment task is a procedure that allows operational transactions to be performed on the peripheral IoT infrastructure. Computing nodes, represented by IoT devices, act as calculators. They receive tasks and must execute them. The central distribution node sends tasks to computing nodes and collects completed tasks. It also monitors the state of communication channels and the state of computing nodes.

A distributed information system built on the basis of the IoT infrastructure is a parallel distributed system [32]. But unlike classical parallel systems, computing nodes are heterogeneous and the switching network changes the characteristics of delay and throughput during operation.

Let us give a formal definition of the task assignment problem to IoT computing nodes. It is based on the problem of optimal mapping of the algorithm onto the architecture of a parallel computing system using a graph model [33]. Let the computing algorithm be represented as an acyclic graph

$$G_A = \left\langle A, C^A \right\rangle, \quad (1)$$

where $A = \left(a_1, a_2, \ldots, a_{card(A)}\right)$ – graph nodes corresponding to the components of the algorithm or tasks; $C^A = \left(c_{i,j}^A, i, j \in \overline{1, |A|}, i \neq j\right)$ – graph edges corresponding to information links between tasks; $c_{i,j}^A$ – amount of information transmitted from the task a_i to the task a_j.

A task can be a fairly large fragment of the source code of a program intended for execution on a parallel system: a subroutine, function, procedure, module, etc.

Let the distributed computing system be represented as a graph

$$G_Q = \left\langle Q, C^Q \right\rangle, \quad (2)$$

where $Q = \left(q_1, q_2, \ldots, q_{card(Q)}\right)$ – graph vertices corresponding to computing nodes; $C^Q = \left(c_{i,j}^Q, i, j \in \overline{1, card(Q)}, i \neq j\right)$ – graph edges corresponding to communication channels between nodes; $c_{i,j}^Q$ – bandwidth of the communication line between computing nodes q_i and q_j without taking into account delays.

A task assignment can be defined as a graph mapping G_A to the graph G_Q. Considering the Cartesian product of sets $A \times Q$ we introduce the mapping binary matrix:

$$Z = \left(z_{i,j}, i \in \overline{1, card(A)}, \; j \in \overline{1, card(Q)}\right). \quad (3)$$

where, if the task is a_i assigned to node q_j, then $z_{i,j} = 1$, otherwise $z_{i,j} = 0$.

The solution to the problem of assignment to computing nodes will be determined by the optimality criterion. Such a criterion is a reflection μ over the matrix Z, and the solution to the problem is the matrix Z^*, such that

$$\min_{Z \in D_z} \mu(Z) = \mu\left(Z^*\right). \quad (4)$$

where D_Z is a set of admissible mapping matrices.

This problem is NP complete, i.e. its solution can be obtained based on a complete enumeration. But it should be taken into account that the characteristics of the computing nodes of a distributed information system based on IoT are constantly changing. As a result, it is necessary to solve problem (4) often. This will lead to the fact that the distributed computing system will be in a constant solution to the optimization problem. Therefore, in the future, we will look for a fast method that will offer a solution to the distribution problem close to the optimal one.

1.2. Public structure of the agent model. The system for supporting peripheral computing IoT is rigidly decentralized. Therefore, the most acceptable is the agent-based model approach to model development [34]. Key elements of such a model:

– agent which is contained in the object, for carrying out a strategy and forming a strategy for the average;

– the environment which is used to describe all those whose agent is used;

– actions they stipulate the agent with each other with environment;

– states which straightens on whirlwind of camp in environment; Skog of the agent to transfer the environment in some cases;

- – rewards which will give a environment medium to the agent reaction; can booty with negative or positive numerical values.

The agent interacts with the environment and receives reactions from it continuously. The agent generally has no information about the environment and is not told what action to take at each step.

The IoT may change under the influence of the Environment agent.

Therefore, each time an agent for the same activity may receive different reward values. If the environment does not change, then the same agent action will always give the same reward.

According to Fig. 1 agent sends its own action at each discrete time $t \in \mathbf{N}$ to Environment in a_t. In response, Environment transitions to the s_{t+1} and generates a new reward r_{t+1}, that is new to the agent.

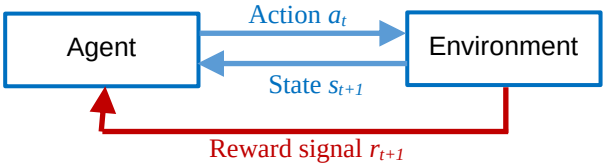

Fig. 1. The model agent – environment

Each state of the environment at each time t is denoted by, where $s_t \in S$, where S – is the set of all possible states. The agent at a particular step t selects an action a_t from a plurality of all actions of the currently possible actions. $A(s_t)$. After that, the environment is in a new state, and the agent receives new environmental data. Thus, the agent learns and accumulates experience. Experience allows the agent to make further decisions regarding the environment - to choose further actions. This means forming a strategy for the behavior of the agent, depending on the current experience.

The agent strategy is described as:

$$\pi_t(a|s) = P(a_t = a / s_t = s). \quad (5)$$

Agents' strategies are aimed at maximizing the total reward over a long period of time.

According to the model, the agent should have:

– a goal related to the state of the environment;

- the possibility of performing some actions that may affect the state of the environment;

- the ability to perceive the state of the environment.

Scheme Fig. 1 is not significantly different from the "agent-environment" scheme for the architecture of neural networks. The main difference is the presence of evaluated feedback. The agent uses information that only evaluates his actions, but does not indicate in any way whether this action is correct or not, since it is impossible to obtain a set of behavior samples that would be correct and cover all possible situations in which the agent may find himself.

The learning process of an agent can be imagined as a set of discrete states between which transitions are made. Taking into account the fact that the agent receives responses from the environment only at discrete moments of time, the environment from the agent's point of view can be considered a discrete system that has states and transitions between them. But we clarify that the environment is not controlled by the agent and may have other influences (besides the agent).

As a result of interaction with the environment, the agent can build its model of this environment. Based on such a model, a prediction algorithm can work for more effective agent work.

In IoT, computing nodes play the role of the environment, the change in characteristics of which, in general, does not have any regularity. Computing nodes are independent of each other, so execution of tasks on one node does not affect other nodes. Also, after completing one task, a computing node enters the pool of free nodes. The completed task does not affect the further functioning of this node. Therefore, taking into account the specified conditions, the best model is a model with an ε-greedy strategy. At the same time, it should be noted that the agent applies only those actions that he knows about and that give the maximum reward. But he also has to consider new actions that may not yield much reward, but might make a better choice for a higher reward in the future.

1.3. Agent's goal and reward. The main goal of an agent in IoT is to maximize the total reward received as a response from the environment and determined by some number.

The reward signal must indicate what the agent needs to achieve. At the same time, the reward signal does not give the agent a priori information about how to achieve the desired result. For example, if the agent searches for the maximum of some function, it receives only the value of this function from the environment (described by the specified function). He has no information about the direction of her growth that he can apply to take the next steps. Otherwise, it will be a reward for achieving intermediate goals. In the example of finding the maximum of a function, the agent can find a local maximum and stop there, never finding a global maximum.

The complete independence of the environment from the agent is also a prerequisite. An agent must not have a goal that it can control (even partially). Otherwise, the agent will be able to independently set the received value of the reward signal.

A sequence of interactions with the environment performed by an agent constitutes a task. Depending on whether or not it is possible to break the task into repeated sequences, the following are distinguished:

– tasks consisting of episodes;

– continuous tasks.

Consider tasks consisting of episodes. Denote through $r_t + 1$, $r_t + 2$, $r_t + 3$, … sequence of reward signal values received after some time step t. It is necessary to determine which elements of this sequence must be maximized. Since the agent's goal is to maximize the total value of all rewards, it is necessary to find the maximum expected benefit for the general case R^*:

$$R^* = \max_{\mu} R(t) = \max_{\mu} \sum_{i=t+1}^{T} r_i(t), \quad (6)$$

where T the final time step, μ is a set of possible options. Formula (6) will be relevant in situations where it is possible to determine the final time step. In this case, the agent's interaction with the environment can be represented as a finite sequence. Such a sequence is called an episode, and each episode ends with a terminal state. After reaching the terminal state, the "agent - environment" system returns to its initial state. The

sequence of episodes constitutes tasks. In such tasks, it is necessary to distinguish the set of all states S from the terminal state S+.

Now let's move on to continuous tasks. In such tasks, the agent's interaction with the environment cannot be represented as a sequence of episodes. Therefore, it is impossible to fix the final moment of time, that is, there is no terminal state.

Since for continuous tasks, the final time step $T = \infty$, then the concept of discounting is introduced. In this case, the present reward is of greater value to the agent than the future reward. For this purpose, the drive factor is introduced (discount rate) $\gamma \in [0; 1]$, which determines the present value of future rewards. Usually, the value of the future reward after *k* time steps decreases (from the current one) by γ^{k-1}.

Thus, instead of maximizing the sum of all rewards (6) of continuous tasks, the agent maximizes the sum of the given rewards it will receive in the future:

$$R^* = \max_{\mu} R(t) = \max_{\mu} \sum_{i=t+1}^{\infty} \gamma^{i-t} r_i(t), \quad \gamma \in [0;1]. \quad (7)$$

When $\gamma = 1$ from (7) formula (6) with the sum of an infinite series is obtained. When $\gamma = 0$ formula (7) is simplified only to the nearest reward:

$$R^* = \max_{\mu} r_{t+1}(t), \quad (8)$$

that is, the agent is only interested in maximizing the nearest reward. In this case, the goal of the agent is to choose an action a_t in such a way as to maximize $r_{t+1}(t)$.

By seeking to maximize only immediate rewards, the agent limits its influence on future rewards. This leads to a decrease in the overall benefit. As the guidance coefficient γ increases (approaching 1), the agent becomes more far-sighted and the importance of future rewards increases.

1.4. Determining the value of an action. The value of a particular action is defined as the sum total of all rewards that the agent can receive in the future starting from that action. Let's determine the actual value of action *a* as $W(a)$. The expected value of this action at the time step t is denoted as $W_t(a)$. In the case of choosing action *a* until the moment *t* is equal k_a times, we will receive a sequence of rewards r_1, r_2, ..., r_k. Then the value of action a can be estimated as follows:

$$\begin{cases} W_t(a) = 0, & if \quad k_a = 0; \\ W_t(a) = \dfrac{1}{k_a} \sum_{i=1}^{k_a} r_i, & if \quad k_a \neq 0. \end{cases} \quad (9)$$

When $k_a \to \infty$ $W_t(a) \to W^*(a)$.

For each type of tasks, it is possible to determine the benefit value separately.

But if, after the last state in the final episode, we add an infinite number of states that will give zero reward, then we can enter an absorbing state, from which the transition will be possible only into itself (Fig. 2).

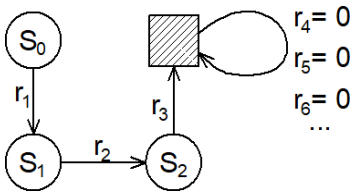

Fig. 2. Absorbing state

The absorbing state allows you to get the same benefit as when summing an infinite sequence. Thus, in the general case, the benefit is described by the equation

$$R(t) = \sum_{k=0}^{T} \gamma^k \cdot r_{t+k+1}, \quad (10)$$

where can be either $T = \infty$, або $\gamma = 1$ (but not at the same time).

1.4.1. IoT stationary environment. In a stationary environment, the reaction of the environment to the actions of the agent does not change over time [35]. Therefore, the assessment of the reward of each action does not change over time.

When the number of time steps increases, the memory requirements of the computer system also increase, because it is necessary to accumulate all the reward values for the entire time of operation. Let's determine for action *a* the average value of its k rewards as W_k. Then:

$$W_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} r_i = \frac{1}{k+1} \left(r_{k+1} + \sum_{i=1}^{k} r_i \right). \quad (11)$$

But $$W_k = \frac{1}{k} \sum_{i=1}^{k} r_i \quad \Rightarrow \quad \sum_{i=1}^{k} r_i = k \cdot W_k. \quad (12)$$

Put (12) in (11):

$$W_{k+1} = \frac{1}{k+1} (r_{k+1} + k \cdot W_k) =$$

$$= \frac{1}{k+1} (r_{k+1} + (k \cdot W_k + W_k) - W_k) = \frac{1}{k+1} \times$$

$$\times (r_{k+1} + W_k (k+1) - W_k) = W_k + \frac{1}{k+1} (r_{k+1} - W_k).$$

We will make a replacement:

$$\frac{1}{k+1} = \frac{1}{k_a + 1} = a_{k+1}(a) = a. \quad (13)$$

Then

$$W_{k+1} = W_k + a(r_{k+1} - W_k), \quad (14)$$

where *a* is step length value.

This recursive formula can be written in the form of a rule:

New rating ← Previous rating +
+ Step length × Error,

where

Error ← [Target – Old assessment].

The error decreases with each step closer to the target, and the “Step Length” parameter changes with each time step.

1.4.2. IoT non-stationary environment. In tasks performed in a non-stationary IoT environment, feedback received from the environment at some point in time better reflects the current situation than feedback received at some earlier time period.

From formula (13), we can see that the value of step a depends on the number of times when some action was selected.

If we set a constant value of step a, then it becomes possible to take into account the non-stationarity of the environment. As a result, we will get a weighted average value W_k :

$$\begin{aligned} W_k &= W_{k-1} + a\left(r_k - W_{k-1}\right) = W_{k-1} + a \cdot r_k - a \cdot W_{k-1} = \\ &= a \cdot r_k + (1-a) W_{k-1} = a \cdot r_k + (1-a) \times \\ &\times \left(W_{k-2} + a\left(r_{k-1} - W_{k-2}\right)\right) = a \cdot r_k + (1-a) \cdot a \cdot r_{k-1} + \\ &+ (1-a)^2 W_{k-2} = \ldots = a \cdot r_k + (1-a) \cdot a \cdot r_{k-1} + \\ &+ \ldots + (1-a)^{k-1} a \cdot r_1 + (1-a)^k W_0 = \\ &= (1-a)^k W_0 + \sum_{i=1}^{k} a \cdot (1-a)^{k-i} \cdot r_i . \end{aligned} \quad (15)$$

The value obtained in (15) is a weighted average, because the following equality holds for the sum of weights:

$$(1-a)^k + \sum_{i=1}^{k} a \cdot (1-a)^{k-i} = 1. \quad (16)$$

Then, the weight value $a \cdot (1-a)^{k-i}$ for reward r_i depends on the size $k - i$, that is, from how many steps ago this reward was received. Since the size $(1 - a) < 1$, then the weight of the reward r_i decreases exponentially with the increase in the number of rewards received.

To ensure sequence convergence, the following conditions must be met:

$$\sum_{i=1}^{k} a_k \cdot (a) = \infty; \quad (17)$$

$$\sum_{i=1}^{k} \left(a_k (a)\right)^2 < \infty. \quad (18)$$

Condition (17) guarantees that the steps have a sufficiently large value so that the learning process is not affected by random fluctuations. Condition (18) guarantees that the steps are small enough to ensure convergence.

Non-stationary tasks also have a significant difference from stationary ones. In them, the initial evaluation of the value of the action does not influence the work results.

1.5. A strategy for choosing the agent's next action. The agent chooses the next action based on the evaluation of the value of the action, using the following behavioral strategies:

– greedy strategy;
– ε - greedy strategy;
– random selection.

The greedy strategy is the simplest variant of the agent's behavior: the agent each time chooses the action that has the greatest impact value assessment.

Therefore, at some time step t, such an action should be chosen a^* , when

$$W_t (a^*) = \max_a W_t (a). \quad (19)$$

In this strategy, the agent never tries to explore the environment in order to discover some action that would yield a higher reward. In a stationary environment, when all reward values are known, this behavior is the most correct, but for a non-stationary environment, this behavior will not always give the best result.

The opposite of a greedy strategy is a strategy based on random selection, where the agent chooses a random value at each step. In this case, regardless of the obtained result, the agent always explores the environment. But the agent never uses the received information to plan its future behavior. Random behavior can result in a non-stationary environment, when the parameters of the environment change randomly. For a non-stationary environment, the most acceptable option is ε - a greedy strategy, where a variable coefficient ε is introduced, specifying the possibility of choosing an action. With such a strategy, the agent alternately chooses either a greedy action or a non-greedy one. This mode does not allow you to get the maximum reward when performing the current action, but it can lead to an increase in the total reward over a long time, that is, lead to the maximization of the amount of rewards.

The ε - greedy strategy can be implemented using the Multi-armed bandit algorithm, where the environment is represented as a slot machine that has not one but many levers with different rewards. In the algorithm, the agent performs some specific action. A reward is obtained from the set A of all actions available to the agent $r \in R$.

To find a suboptimal solution in the algorithm, it is necessary to ensure a balance between the research process and the exploitation process. The MAB algorithm uses greedy and ε -greedy strategies.

With a greedy strategy, the algorithm selects only the action with the largest known reward. This gives the maximum reward at this point in time, but does not take into account that the possible rewards for other actions may increase over time. With an ε-greedy strategy, the coefficient ε is in the range $\varepsilon \in [0; 1]$ and regulates the strategy of the agent's behavior - to act completely "greedy" or to periodically randomly choose other actions with probability ε. At ε = 0 strategy becomes greedy; at ε = 1 the algorithm becomes random.

Taking into account the above, the greedy algorithm can be used to assign tasks to IoT computing nodes, where the distributing node acts as the agent, and the computing nodes that perform the received tasks and return the result to the agent act as the environment.

2. The computing node model

IoT devices are represented by a variety of technologies and components and their constituents. However, a general description can be made for IoT

devices, which will provide an abstraction from the implementation features and remove possible restrictions on interaction in an IoT distributed system (IoT DS). The general description is based on the identification of general characteristics of IoT devices, namely:

– data on the state of IoT devices: computing power, RAM size, processor frequency, response time, service time, amount of energy consumed, etc.;

– data on the data transmission network of the environment where the IoT device operates: channel capacity, loss indicators, data transfer rate, etc.;

– data on the location of the IoT device and its movement: positioning in space, distance to other IoT computing nodes, etc. Based on these characteristics, it is possible to determine the ability of a computing node to assign a task to it in the IoT DS, taking into account the reinforcement machine learning model in the form:

$$Reward = (State, Location, Network), \quad (20)$$

where *State* is the state data, *Network* is the network, *Location* is the position and movement.

The computing nodes of the IOT DS act as the environment. It should form a reaction by means of a reward signal - *Reward*.

Accordingly, each computing node in the IoT DS forms the Reward parameter. This parameter absorbs all the characteristics, dependent and independent of the computing node, and is calculated by the computing node as their final integral characteristic. Thus, the parameter reflects the ability and readiness to assign a task to the computing node of the IoT DS.

However, since the types of tasks can change, the type of task solved by the computing node should also be taken into account. Then the model of the computing node is defined as:

$$D = (ID, Label, Reward), \quad (21)$$

where D – IoT device, ID – node identifier (unique number); *Label* – type of solved task; *Reward* – the value of the reward signal.

This model of the IoT DS computing node describes the readiness of a specific computing node to accept the next task.

Let's consider how to define the reward function. Consider a separate computing node as an element of the environment. Then it can be argued that the value of the reward is determined by the ability of the node to successfully perform the task received. The success of the task depends on a number of parameters of both the node itself and the parameters of other IoT DS elements. Parameters dependent on the computing node are as follows [36]:

– frequency and operating modes of the processor;

– amount and speed of RAM;

– PZP speed (if used);

– architectural solutions of IoT devices;

– IoT device software (operating system, presence or absence of additional software, etc.);

– number of sensors, methods and sequence of their survey;

– availability of additional peripheral equipment and its parameters.

The parameters that do not depend on the compute node are as follows:

– the width of the communication channel;

– bandwidth of the communication channel;

– the presence of obstacles in the communication channel;

– used network protocols;

– parameters of the distribution node;

– remoteness of IoT devices from IoT DS elements;

– the presence of the IoT device in a static or dynamic state.

The above lists of parameters are quite complete, but not exhaustive. Thus, the reward is an integral characteristic of the node, which must take into account all of the above, as well as, if necessary, other parameters).

In general, these characteristics are summarized in the Reward parameter as follows:

$$Reward = \sum_{i=1}^{n} p_i \cdot \lambda_i, \quad (22)$$

where p_i – value of the i-th parameter; λ_i – weighting factor; n – the number of parameters affecting the value of the reward signal.

It is often impossible to determine the number of parameters and the degree of their influence on the overall value of the reward.

It is also necessary to take into account that the parameters change at every moment of time. In such cases, the reward function is proposed to be used as a function of the task's cycle time (t_{RTT}). That is, it is possible to define the task cycle time as the time spent sending the task to the computing node, the time to execute the task on the computing node, and the time to receive the response from the computing node. Thus, the circulation time can be calculated using the formula:

$$t_{RTT} = t_{answer} - t_{sending}, \quad (23)$$

where t_{answer} – the time of receiving a response from the computing node; $t_{sending}$ – the time of sending the task by the distributing node.

Then the reward will be defined as a function:

$$Reward = Reward(t_{RTT}), \quad (24)$$

which allows you to take into account all the parameters of the DS IoT that can affect the success of the task. At the same time, the computing node can also send its reward value, which can be used by the distributing node to calculate the total reward value, taking into account the circulation time.

3. Algorithm for distribution of tasks by computing nodes

3.1. The main algorithm. Based on the agent's work with the environment, we will develop an algorithm for implementing the distribution of tasks by computing nodes.

Consider a multi-agent system, where DS IoT elements can be represented as a set of interacting agents.

But DS IoT is implemented on devices with limited computing capabilities and with communication channels of low bandwidth. Therefore, it is proposed to implement the task distribution algorithm on the distribution node.

In addition, during the exploitation phase, the distributing node receives a response from the computing node, which also contains the value of the current reward along with the result.

This approach allows to reduce the amount of data transmitted by the network due to the elimination of additional polling of computing nodes.

Let's consider the task assignment algorithm step by step.

Initialization stage.

Step 0. The task sequence is initialized by the distributing node in IoT DS.

Research stage.

Step 1. All computing nodes receive a request from the distributing node to receive the integral characteristics of Reward from them. Each value is the value of the current reward signal for each computing node and is constructed based on its characteristics.

Step 2. The integrated characteristics of Reward obtained by the distributing node are transformed into probability values. These probability values are used when selecting a computing node for task assignment.

Operation stage.

Step 3. The distribution node sends the task to the computing nodes. At the same time, it tries to get the maximum value of the reward signal and maximize the total reward over a long period of time.

Step 4. Upon receiving the completed tasks, the distributing node recalculates the value of the reward signals (since the completed tasks also transmit information about the current state of the IoT DS and computing nodes).

Step 5. If there are no outstanding tasks in the sequence of tasks, then the algorithm stops, otherwise it goes to step 3 (for the steady state) or to step 1 (for the non-stationary state).

The general scheme of interaction, according to the above algorithm, looks as shown in Fig. 3.

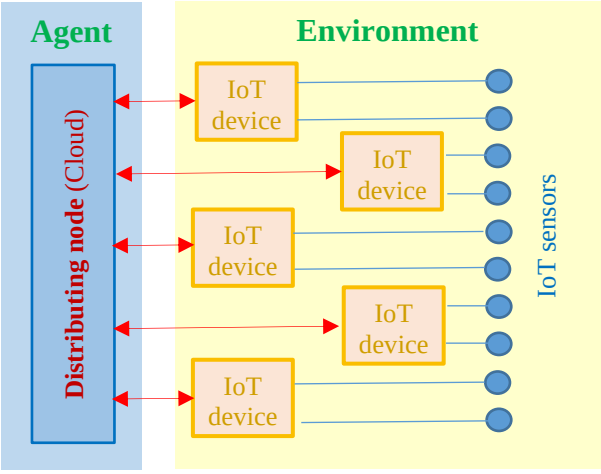

Fig. 3. General scheme of interaction of IoT DS components in the task assignment algorithm

3.2. Modification of the algorithm. Usually, in IoT systems, the distribution node is located in the cloud. Computing nodes that make up a distributed system for performing simple operational tasks are chosen from devices of fog and boundary layers. These devices sit next to IoT sensors. But sometimes a significant percentage of the time from the request for task processing to receiving the result is spent exchanging with the cloud, which is unacceptable when performing operational tasks. Based on this, a modification of the main algorithm is proposed. The meaning of the modification is to reduce the number of requests to the cloud.

Therefore, it is necessary to organize a check of the ability of computing nodes to assign tasks with minimal participation of the distribution node. For this purpose, it is proposed to define clusters of devices with similar characteristics.

Clustering methods can be divided into two main types: hard and soft. The first type of methods, unlike the second, has clear boundaries, which is unacceptable for dynamic computing nodes. The principle of soft clustering assumes that a computer node can belong to one or several clusters at the same time. The application of this principle to the computing nodes of IoT DS W is justified, since the characteristics of IoT devices usually change.

Therefore, one device can fall into several clusters simultaneously based on its variable characteristics.

Among the existing algorithms for performing clustering, the Fuzzy C-Means (FCM) clustering algorithm stands out, which is based on data on the similarity of sets and is a soft clustering algorithm. Clustering of computing nodes was performed using this algorithm under the following conditions:

– the number of clusters is finite and constant;

– the centroid of each cluster is calculated according to the following formulas:

$$p_{jk} = \frac{1}{\sum_{i=1}^{n} \left(d_{jk} / d_{ik} \right)^{2/(m-1)}}, \qquad (25)$$

$$q_k = \sum_{j=1}^{n} p_{jk}^m w_j \Big/ \sum_{j=1}^{n} p_{jk}^m, \qquad (26)$$

де i, j – device numbers IoT, $i, j \in \overline{1,n}$; w_j – coordinate of the j-th device IoT; k – cluster number; d_{jk} – the distance from the IoT device to the centroid according to the Euclidean metric; m – the fuzziness index, which is a parameter of the fuzziness region, usually $m = 2$; p_{jk} – the calculated probability of the j-th IoT device belonging to the k-th cluster; q_k – the current coordinate of the centroid of the k-th cluster.

The algorithm allows you to determine the degree of belonging of each IoT device to each cluster.

In this case, the computer node that has the sending task acts as an agent. With the help of the developed algorithm, it searches for the optimal node for

assigning the task and becomes the distributing node itself. All other computing nodes relative to it are the external environment.

The modified version of the developed algorithm consists of a sequence of the following steps:

Initialization stage.

Step 0. Initialization of the sequence of tasks and initial values in DS IoT.

Research stage.

Step 1. The distributing node polls all computing nodes in DS IoT and receives from each the Reward value generated by the computing node according to the computing node model. The resulting Reward values are normalized and transformed by the distributing node.

Step 2. Based on the value of Reward, each computing node is mapped to a distributing node of a cluster or several clusters using the Fuzzy C-Means algorithm. The distribution node stores the distribution of all computing nodes by cluster. In case of changes, the cluster sets on the distribution node are updated. However, all computing nodes located within each cluster are notified of the presence of other nodes in this cluster.

Step 3. To obtain data about the result of clustering, each computing node sends a request to the distributing node. In return, it receives a record of its cluster(s) and the nodes in it (or in them). If one of the characteristics of the node changes, the transition to *Step* 2 is carried out.

The stage of exploitation.

Step 4. If there is a computing task to be assigned, the computing node assumes the role of the distributing node and chooses another computing node within the cluster. In the case of a node with a task belonging to several clusters at the same time, priority is given to the nodes of the cluster that have the highest degree of belonging to this cluster. Thus, the computing node for some time becomes the distributing node that assigns tasks. The further sequence of steps is performed according to the main task assignment algorithm.

Step 5. Sending reward values by computing nodes to the distribution node within the cluster.

Step 6. Calculation of the optimal node for task assignment. If there is no possibility to assign a task, the transition to Step 2 takes place.

Step 7. Assignment of the task to the node.

The cluster selection architecture is shown in Fig. 4.

4. Discussion of results

The features of the main task assignment algorithm remain the same as before the modifications made, since these modifications do not affect the target task assignment model. The changes only allow redefining the participants in the task assignment process. Thanks to this, the possibilities for applicants for the role of a distribution node in DS IoT are expanding. Thus, the state of DS IoT computing nodes can be either static or dynamic. So, the advantage of the task allocation algorithm is its versatility, because the same algorithm can work with both static and dynamic environments. Using the coefficient ε, you can change the behavior of the algorithm and better adjust it to the computing environment with which the distribution node works.

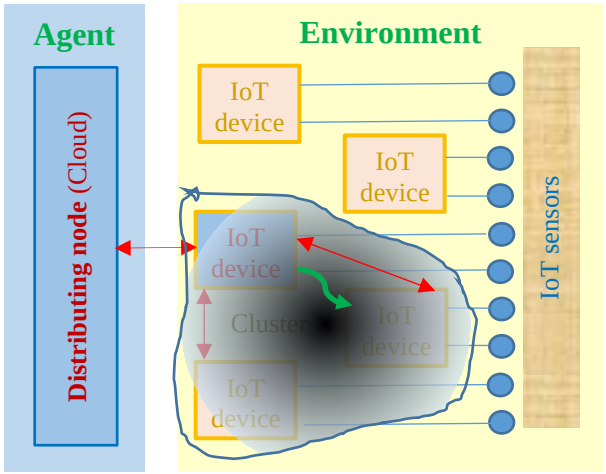

Fig. 4. The general diagram of the interaction of the DS IoT components in the modified task assignment algorithm

If the state of the computing nodes does not change, then it makes no sense to conduct additional checks for the variability of the environment. Then the value of the parameter ε should approach 0, but not reach it. Since the probability of choosing each computing node is not equal to zero, the situation will not arise that the algorithm allocates tasks only to the nodes that have the highest value of the reward signal.

In the case of significant variability of the computing environment, it makes sense to make the value of the parameter ε as close as possible to 1. This allows you to constantly explore new computing nodes, since their parameters could improve in a short time. Setting the parameter ε close to 1 in the case of a static environment does not significantly affect the task allocation process. Setting the parameter ε close to 0 for a dynamic environment can have catastrophic consequences for the computational process, even to the point of its complete impossibility.

A situation is possible when the state of the data transmission network and computing nodes is unknown. Then you can select the ε parameter during DS IoT operation.

It is necessary to assume from the beginning that the system is dynamic with significant changes in parameters, that is, ε will be close to unity. Then, in the process of executing the work, the distributing node can accumulate statistics about the changes in the reward signal of each computing node.

Accordingly, it is necessary to reduce the value of the parameter ε until the performance of the system reaches its maximum value.

Next, we will investigate the behavior of the algorithm in different states. In this study, we will assume:

– one distribution node in DS IoT;

– only computing nodes in DS IoT are available to the distributing node in DS IoT, additional actions are not entered in DS IoT;

– during the operation of the algorithm on all iterations, the distributing node does not change;

– the number of computing nodes in DS IoT is constant and does not change over time;

– the reward signal of computing nodes in DS IoT may change;

The following options were considered:

V0 – static;

Vj – d = 0.1 + 0.2 · j, where d – the probability of changing the characteristics of the computing node, j ∈ {0, 1, 2. 3, 4}.

The simulation results are shown in table. 1, where the average execution time of a batch of consecutive operational transactions is given in seconds.

Table 1 – **Execution time of a batch operational transactions**

ε \ V	V0	V1	V2	V3	V4	V5
0.01	**48**	50	47	52	94	183
0.1	49	49	46	54	77	92
0.2	54	**45**	51	47	72	85
0.3	57	48	**42**	41	61	82
0.4	56	46	46	42	44	62
0.5	62	52	51	**39**	46	44
0.6	63	50	53	41	39	48
0.7	71	56	50	44	**34**	33
0.8	66	59	46	40	36	30
0.9	72	65	51	41	35	31
0.99	75	66	48	45	41	**28**

As we can see, the simulation results confirmed the conclusions made on the modified algorithm for distributing tasks across computing nodes.

Conclusions

Based on the analysis of the IoT objects and distributed computing systems, a conclusion was made about the possibility of constructing a distributed information system based on the IoT devices. Formalization of the task distribution process allows us to approach the consideration of a computational problem in the form of a graph.

This graph is transformed into a sequence of tasks sent to the computing nodes of the IoT distributed information system.

A model of a computing node was formed, which made it possible to specify a separate computing node, taking into account its location and functioning features.

Particular attention was paid to the value of the reward signal that the computing nodes will send to the distributing node to implement the task distribution algorithm.

The reward signal is an integral characteristic of the computing node and can depend on many parameters of the DS IoT.

It is proposed to use the time from sending the task by the distributing node to receiving the results from the computing node to calculate the reward.

A method for distributing tasks among the nodes of a distributed information system was developed. The method allows taking into account the features of each computing node and the state of communication channels between them.

Based on the analysis of a stationary or non-stationary environment and changing the greedy strategy of one agent and a set of actions, it became possible to build an algorithm for distributing tasks.

Acknowledgements

The study was funded by the National Research Foundation of Ukraine in the framework of the research project 2022.01/0017 on the topic “Development of methodological and instrumental support for Agile transformation of the reconstruction processes of medical institutions of Ukraine to overcome public health disorders in the war and post-war periods”.

References

1. Fabre, W., Haroun, K., Lorrain, V., Lepecq, M. and Sicard, G. (2024), “From Near-Sensor to In-Sensor: A State-of-the-Art Review of Embedded AI Vision Systems”, *Sensors*, vol. 24(16), 5446, doi: https://doi.org/10.3390/s24165446
2. Schulz, A.S. (2023), “User Interactions with Internet of Things (IoT) Devices in Shared Domestic Spaces”, *ACM International Conference Proceeding Series*, pp. 577–579. doi: https://doi.org/10.1145/3626705.3632615
3. Sharma, A. and Singh, N. (2022), “Sensors, Embedded Systems, and IoT Components”, *Mathematical Modeling for Intelligent Systems: Theory, Methods, and Simulation*, pp. 1–15, doi: 10.1201/9781003291916-1
4. Pardo, C., Wei, R. and Ivens, B.S. (2022), “Integrating the business networks and internet of things perspectives: A system of systems (SoS) approach for industrial markets”, *Industrial Marketing Management*, vol. 104, pp. 258–275, doi: https://doi.org/10.1016/j.indmarman.2022.04.012
5. Dotsenko, N., Chumachenko, I., Galkin, A., Kuchuk, H. and Chumachenko, D. (2023), “Modeling the Transformation of Configuration Management Processes in a Multi-Project Environment”, *Sustainability (Switzerland)*, Vol. 15(19), 14308, doi: https://doi.org/10.3390/su151914308
6. Krishnan, S. and Ilmudeen, A. (2023), “Internet of Medical Things in Smart Healthcare: Post-COVID-19 Pandemic Scenario”, *Imprint Apple Academic Press*, New York, doi: http://dx.doi.org/10.1201/9781003369035
7. Zhang, Z. (2023), “A computing allocation strategy for Internet of things’ resources based on edge computing”, *International Journal of Distributed Sensor Networks*, vol. 17(12), doi: https://doi.org/10.1177/15501477211064800
8. Chalapathi, G.S.S., Chamola, V., Vaish, A. and Buyya, R. (2022), “Industrial internet of things (Iiot) applications of edge and fog computing: A review and future directions”, *Advances in Information Security*, vol. 83, pp. 293–325, doi: https://doi.org/10.1007/978-3-030-57328-7_12

9. Fatlawi, A., Al Dujaili, M.J. (2023), Integrating the Internet of Things (IoT) and Cloud Computing Challenges and Solutions: A Review. AIP Conference Proceedings, 2977(1), 020067. doi: http://dx.doi.org/10.1063/5.0181842
10. Qayyum, T., Trabelsi, Z., Waqar Malik, A. and Hayawi, K. (2022), "Mobility-aware hierarchical fog computing framework for Industrial Internet of Things", *Journal of Cloud Computing*, vol. 11(1), doi: https://doi.org/10.1186/s13677-022-00345-y
11. Kuchuk, H. and Malokhvii, E. (2024), "Integration of IOT with Cloud, Fog, and Edge Computing: A Review", *Advanced Information Systems*, vol. *8*(2), pp. 65–78, doi: https://doi.org/10.20998/2522-9052.2024.2.08
12. Kuchuk, N., Mozhaiev, O., Semenov, S., Haichenko, A., Kuchuk, H., Tiulieniev, S., Mozhaiev, M., Davydov, V., Brusakova, O. and Gnusov, Y. (2023). Devising a method for balancing the load on a territorially distributed foggy environment. Eastern-European Journal of Enterprise Technologies, vol. 1(4 (121), pp. 48–55, doi: https://doi.org/10.15587/1729-4061.2023.274177
13. Hunko, M., Tkachov, V., Kuchuk, H. and Kovalenko, A. (2023), Advantages of Fog Computing: A Comparative Analysis with Cloud Computing for Enhanced Edge Computing Capabilities, *2023 IEEE 4th KhPI Week on Advanced Technology, KhPI Week 2023 – Conf. Proc*, 02-06 October 2023, Code 194480, doi: https://doi.org/10.1109/KhPIWeek61412.2023.10312948
14. Lu, S., Wu, J., Wang, N., Duan, Y., Liu, H., Zhang, J. and Fang, J. (2023), "Resource provisioning in collaborative fog computing for multiple delay-sensitive users", *Software – Practice and Experience*, vol. 53, is. 2, pp. 243–262, doi: https://doi.org/10.1002/spe.3000
15. Kuchuk, G., Nechausov, S. and Kharchenko, V. (2015), "Two-stage optimization of resource allocation for hybrid cloud data store", *Int. Conf. on Information and Digital Techn*, Zilina, pp. 266–271, doi: http://dx.doi.org/10.1109/DT.2015.7222982
16. Petrovska, I. and Kuchuk, H. (2023), "Adaptive resource allocation method for data processing and security in cloud environment", *Advanced Information Systems*, vol. 7, no. 3, pp. 67–73, doi: https://doi.org/10.20998/2522-9052.2023.3.10
17. Li, G., Liu, Y., Wu, J., Lin, D. and Zhao, Sh. (2019), "Methods of Resource Scheduling Based on Optimized Fuzzy Clustering in Fog Computing", *Sensors*, MDPI, vol. 19(9), doi: https://doi.org/10.3390/s19092122
18. Jamil, B. Shojafar, M., Ahmed, I., Ullah, A., Munir, K. and Ijaz, H. (2020), "A job scheduling algorithm for delay and performance optimization in fog computing", *Concurrency and Computation: Practice and Experience*, vol. 32(7), doi: https://doi.org/10.1002/cpe.5581
19. Gomathi, B., Saravana Balaji, B., Krishna Kumar, V., Abouhawwash, M., Aljahdali, S., Masud, M. and Kuchuk, N. (2022), "Multi-Objective Optimization of Energy Aware Virtual Machine Placement in Cloud Data Center", Intelligent Automation and Soft Computing, Vol. 33(3), pp. 1771–1785, doi: http://dx.doi.org/10.32604/iasc.2022.024052
20. Proietti Mattia, G. and Beraldi, R. (2023), "P2PFaaS: A framework for FaaS peer-to-peer scheduling and load balancing in Fog and Edge computing", *SoftwareX*, vol. 21, doi: https://doi.org/10.1016/j.softx.2022.101290
21. Kuchuk, N., Kovalenko, A., Ruban, I., Shyshatskyi, A., Zakovorotnyi, O. and Sheviakov, I. (2023), "Traffic Modeling for the Industrial Internet of NanoThings", *2023 IEEE 4th KhPI Week on Advanced Technology*, KhPI Week 2023 - Conference Proceedings, 2023, doi: 194480. http://dx.doi.org/10.1109/KhPIWeek61412.2023.10312856
22. Kuchuk, H., Kalinin Ye., Dotsenko N., Chumachenko I. and Pakhomov Yu. (2024), "Decomposition of integrated high-density IoT Data Flow", *Advanced Information Systems*, vol. 8, no. 3, pp. 77–84, doi: https://doi.org/10.20998/2522-9052.2024.3.09
23. Attar, H., Khosravi, M.R., Igorovich, S.S., Georgievan, K.N. and Alhihi, M. (2020), "Review and performance evaluation of FIFO, PQ, CQ, FQ, and WFQ algorithms in multimedia wireless sensor networks", *International Journal of Distributed Sensor Networks*, vol. 16(6), doi: https://doi.org/10.1177/1550147720913233
24. Sharma, Sh. Saini H. (2019), "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment", *Sustainable Computing: Informatics and Systems*, vol. 24, doi: https://doi.org/10.1016/j.suscom.2019.100355
25. Liu, L., Chen, H., Xu, Z. (2022). SPMOO: A Multi-Objective Offloading Algorithm for Dependent Tasks in IoT Cloud-Edge-End Collaboration. Information, 13, 75. doi: https://doi.org/10.3390/info13020075
26. Malik, U.M., Javed, M.A., Frnda, J., Rozhon, J. and Khan, W.U. (2022), "Efficient Matching-Based Parallel Task Offloading in IoT Networks", *Sensors*, vol. 22, doi: https://doi.org/10.3390/s22186906
27. Kuchuk, G.A., Akimova, Yu.A. and Klimenko, L.A. (2000), "Method of optimal allocation of relational tables", *Engineering Simulation*, 2000, vol. 17(5), pp. 681–689, available at: https://www.scopus.com/record/display.uri?eid=2-s2.0-0034512103&origin=resultslist&sort=plf-f#metrics
28. Ghenai, A., Kabouche, Y. and Dahmani, W. (2018), "Multi-user dynamic scheduling-based resource management for Internet of Things applications", *2018 International Conference on Internet of Things, Embedded Systems and Communications* (IINTEC), doi: https://doi.org/10.1109/IINTEC.2018.8695308
29. Wei, J.-Y. and Wu, J.-J. (2023), "Resource Allocation Algorithm in Industrial Internet of Things Based on Edge Computing", *Journal of Northeastern University*, vol. 44(8). doi: https://doi.org/10.12068/j.issn.1005-3026.2023.08.002
30. Yaloveha, V., Podorozhniak, A. and Kuchuk, H. (2022), "Convolutional neural network hyperparameter optimization applied to land cover classification", *Radioelectronic and Computer Systems*, vol. 1(2022), pp. 115–128, doi: https://doi.org/10.32620/reks.2022.1.09
31. Zhang, Z. (2023), "A computing allocation strategy for Internet of things' resources based on edge computing", *International Journal of Distributed Sensor Networks*, vol. 17(12), doi: https://doi.org/10.1177/15501477211064800
32. Petrovska, I., Kuchuk, H., Kuchuk, N., Mozhaiev, O., Pochebut, M. and Onishchenko, Yu. (2023), "Sequential Series-Based Prediction Model in Adaptive Cloud Resource Allocation for Data Processing and Security", *2023 13th International Conference on Dependable Systems, Services and Technologies, DESSERT 2023*, 13–15 October, Athens, Greece, code 197136, doi: https://doi.org/10.1109/DESSERT61349.2023.10416496
33. Kalinin, Y., Kozhushko, A., Rebrov, O. and Zakovorotniy, A. (2022), "Characteristics of Rational Classifications in Game-Theoretic Algorithms of Pattern Recognition for Unmanned Vehicles", *2022 IEEE 3rd KhPI Week on Advanced Technology* (KhPIWeek), Kharkiv, Ukraine, pp. 1-5, doi: https://doi.org/10.1109/KhPIWeek57572.2022.9916454

34. Shakya, J., Chopin, M. and Merghem-Boulahia, L. (2024), “D,ynamic Coalition Formation among IoT Service Providers: A Systematic Exploration of IoT Dynamics Using an Agent-Based Model”, *Sensors*, vol. 24(11), no. 3471, doi: https://doi.org/10.3390/s24113471
35. Aburukba, R.O., Landolsi, T. and Omer, D. (2021), “A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices”, *Journal of Network and Computer Applications*, vol. 180, no. 102994, doi: https://doi.org/10.1016/j.jnca.2021.102994
36. Li, W., Zhao, B., Zhu, L., Yixuan W., Zhong, Q. and Yu, S. (2024), “TCEC: Integrity Protection for Containers by Trusted Chip on IoT Edge Computing Nodes”, *IEEE Sensors Journal*, doi: https://doi.org/10.1109/JSEN.2024.3445576

Received (Надійшла) 26.07.2024
Accepted for publication (Прийнята до друку) 16.10.2024

ВІДОМОСТІ ПРО АВТОРІВ/ ABOUT THE AUTHORS

Кучук Ніна Георгіївна – доктор технічних наук, професор, професорка кафедри обчислювальної техніки та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Nina Kuchuk – Doctor of Technical Sciences, Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: nina_kuchuk@ukr.net; ORCID Author ID: http://orcid.org/0000-0002-0784-1465;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57196006131.

Кашкевич Світлана Олександрівна – старший викладач кафедри Інтелектуальних кібернетичних систем, Національний авіаційний університет, Київ, Україна;
Svitlana Kashkevich – Senior Lecturer of the Department of Intelligent Cybernetic Systems, National Aviation University, Kyiv, Ukraine;
e-mail: svitlana.kashkevych@npp.nau.edu.ua; ORCID Author ID: https://orcid.org/0000-0002-4448-3839;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=58244269900.

Радченко В'ячеслав Олексійович – старший викладач кафедри Електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Viacheslav Radchenko – Senior Lecturer of Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: viacheslav.radchenko@nure.ua; ORCID Author ID: https://orcid.org/0000-0001-5782-1932;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57189376280.

Андрусенко Юлія Олександрівна – асистент кафедри Електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Yuliia Andrusenko – Assistant Professor of Department of Electronic Computers, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: yuliia.andrusenko@nure.ua; ORCID Author ID: https://orcid.org/0000-0001-7844-2042.

Кучук Георгій Анатолійович – доктор технічних наук, професор, професор кафедри комп’ютерної інженерії та програмування, Національний технічний університет “Харківський політехнічний інститут”, Харків, Україна;
Heorhii Kuchuk – Doctor of Technical Sciences, Professor, Professor of Computer Engineering and Programming Department, National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine;
e-mail: kuchuk56@ukr.net; ORCID Author ID: http://orcid.org/0000-0002-2862-438X;
Scopus ID: https://www.scopus.com/authid/detail.uri?authorId=57057781300.

Застосування периферійних обчислень при виконанні оперативних транзакцій IoT

Н. Г. Кучук, С. О., Кашкевич, В. О. Радченко, Ю. О. Андрусенко, Г. А. Кучук

Анотація. Актуальність. Обробка інформації IoT зазвичай виконується у хмарному середовищі. Але при цьому виникають проблеми, пов’язані з затримками при передачі даних до хмари. Особливо важливо зменшити ці затримки при обробці оперативних транзакцій IoT. Це можливо здійснити за рахунок перенесення частини обчислень на периферійні пристрої IoT. Але при цьому треба враховувати специфічні особливості вбудованих систем IoT. **Предметом вивчення** в статті є методи перенесення навантаження на периферійні пристрої IoT. **Метою статті** є зменшення часу виконання оперативних транзакцій IoT за рахунок підвищення ефективності інфраструктури системи шляхом перенесення частини обчислювального навантаження на периферійні пристрої IoT. Отримано **такі результати**. Зроблено висновок про можливість побудови розподіленої інформаційної системи на основі пристроїв Інтернету речей. Сформована модель обчислювального вузла, яка дозволила задавати окремий обчислювальний вузол, враховуючи його особливості розташування та функціонування. Розроблений метод розподілу завдань по вузлам розподіленої інформаційної системи. Метод дозволяє враховувати особливості кожного обчислювального вузла і стан каналів зв'язку між ними. Розроблений алгоритм реалізації методу базується на аналізі стаціонарного або нестаціонарного середовища та зміни ε-жадібної стратегії агента. **Висновок.** Проведено дослідження ефективності запропонованого. Результати моделювання показали, що запропонований метод дозволяє суттєво зменшити час обробки оперативних транзакцій.

Ключові слова: Інтернет речей; комп’ютерна система; хмарний шар; периферійні обчислення; оперативна транзакція.