UDC 004.41:519.7
DOI: 10.15587/1729-4061.2025.326040

DEVISING A METHOD FOR STABILIZING CONTROL OVER A LOAD ON A CLUSTER GATEWAY IN THE INTERNET OF THINGS EDGE LAYER

Heorhii Kuchuk
Doctor of Technical Sciences, Professor*
Oleksandr Mozhaiev
Corresponding author
Doctor of Technical Sciences, Professor**
E-mail: mozhaev1957@gmail.com
Serhii Tiulieniev
PhD
National Scientific Center "Hon. Prof. M.S. Bokarius Forensic Science Institute"
Zolochivska str., 8, Kharkiv, Ukraine, 61177
Mykhailo Mozhaiev
Doctor of Technical Sciences***
Nina Kuchuk
Doctor of Technical Sciences, Professor*
Liliia Tymoshchyk
PhD***
Andrii Lubentsov
PhD***
Yurii Gnusov
PhD, Associate Professor**
Sergii Klivets
PhD
Science Center
Ivan Kozhedub Kharkiv National Air Force University
Sumska str., 77/79, Kharkiv, Ukraine, 61023
Alexander Kuleshov
PhD, Associate Professor
Science Center
Ivan Kozhedub Kharkiv National Air Force University
Sumska str., 77/79, Kharkiv, Ukraine, 61023
*Department of Computer Engineering and Programming
National Technical University "Kharkiv Polytechnic Institute"
Kyrpychova str., 2, Kharkiv, Ukraine, 61002
**Department of Cyber Security and DATA Technologies
Kharkiv National University of Internal Affairs
L. Landau ave., 27, Kharkiv, Ukraine, 61080
***Scientific Research Center for Forensic Expertise in the Field of Information Technologies and Intellectual Property of the Ministry of Justice of Ukraine
L. Ukrainka blvd., 26, Kyiv, Ukraine, 01133

The object of this study is the process of managing overload at the boundary layer of the geographically distributed Internet of Things.

The task addressed is reducing the number of losses of information packets of the geographically distributed Internet of Things arriving at the boundary layer gateway. For this purpose, it was proposed to use fast temporal horizontal scaling and stabilizing control over the load formed in the gateway buffer.

In the process of conducting research, a temporal horizontal scaling algorithm was developed for the boundary layer cluster gateway. The evolutionary Firefly Algorithm with a fitness function based on the Lorenz function was used in the development. This made it possible to speed up the search for a cluster node for operational temporal scaling of the gateway for the period of gateway buffer overload.

The standard algorithm for intelligent queue management has been modified. The modification is based on the proposed method for stabilizing load control over the boundary layer cluster gateway of the geographically distributed Internet of Things. The method takes into account the features of the boundary layer architecture. The proposed method made it possible, in case of reaching the upper threshold of the queue, to scale the gateway until its buffer is filled. As a result, the number of information packet losses arriving at the boundary layer gateway was reduced. Studies of the proposed method showed that the number of information packet losses is reduced compared to existing methods. The research results can be explained by the use of temporary horizontal scaling of the gateway and fixing the thresholds of the information packet queue buffer. The method is effective at an average load level on the cluster gateway from 0.2 to 1.2

Keywords: Internet of Things, stabilizing control, information packet, buffer overload, boundary calculations

Received 07.01.2025
Received in revised form 04.03.2025
Accepted date 24.03.2025
Published date 29.04.2025

How to Cite: *Kuchuk, H., Mozhaiev, O., Tiulieniev, S., Mozhaiev, M., Kuchuk, N., Tymoshchyk, L., Lubentsov, A., Gnusov, Y., Klivets, S., Kuleshov, A. (2025). Devising a method for stabilizing control over a load on a cluster gateway in the internet of things edge layer. Eastern-European Journal of Enterprise Technologies, 2 (9 (134)), 24–32. https://doi.org/10.15587/1729-4061.2025.326040*

1. Introduction

Geographically Distributed Internet of Things (GDIoT) has an architecture in which sensors, devices, and computing nodes are located over a large area [1]. GDIoT sensors are collectively considered as a wireless sensor network (WSN) [2]. A GDIoT WSN is a collection of various wireless nodes that are sensor-like in nature. These nodes are deployed over a large

Copyright © 2025, Authors. This is an open access article under the Creative Commons CC BY license

geographical area using a gateway [3]. Wireless sensing and data transmission are two major applications of WSN. The widespread application of WSN has affected many industries. Despite its advantages, WSN also has a number of problems, such as resource shortage, energy consumption, memory limitations, and computing power limitations. Of particular note is the issue of possible network overload at the GDIoT wireless sensor network gateway. Some of the side effects of overload are degraded quality of service for GDIoT packets, queuing delays, packet loss, and blocking of re-arrived packets.

GDIoT, like the Internet of Things (IoT), is based on cloud computing technology [4]. However, in recent years, difficulties have arisen due to the following factors [5]:

– the presence of geographical distribution of GDIoT components;

– increased network latency, especially at edge gateways;

– the presence of mobile GDIoT devices.

Most of these difficulties have been solved by introducing additional edge computing layers: Fog and Edge.

The closest to WSN is the GDIoT edge layer. Edge computing is a decentralized subnet of GDIoT nodes and gateways designed to quickly process information packets. Edge computing is a promising concept that brings data processing as close as possible to the end devices of the networks [6]. Data arriving at the edge layer is processed locally, transmitting only important information. Edge layer nodes provide data processing under a mode close to real-time, which is a primary need to minimize latency.

The edge layer of the geographically dispersed Internet of Things has a distributed decentralized structure. The GDIoT edge layer is divided into a number of clusters built on a territorial principle [7]. For each cluster, IoT information packets arrive through an edge layer gateway. Typically, a GDIoT cluster has one gateway to receive and forward information packets from various GDIoT devices. Overloading such a gateway can cause serious problems, including reduced performance, data loss, and increased latency. The causes of overload include:

– excessive data flow, if a large number of IoT devices generate a significant amount of traffic;

– limited computing resources of the gateway;

– network delays and insufficient bandwidth due to insufficient data exchange speed between GDIoT devices and the gateway;

– high load on the gateway due to processing of information packets on it, if the gateway performs complex calculations instead of transmitting data further.

Therefore, the issue of reducing the impact of overloading causes on the gateway of the boundary layer cluster of the geographically dispersed Internet of Things is relevant. Solving this issue would reduce the number of information packet losses and increase the efficiency of GDIoT operation.

2. Literature review and problem statement

In [1], geographically distributed IoT for real-time data collection and experiment management was investigated. However, in this system, the gateways of territorial clusters are loaded by no more than 10 %. Therefore, in this case, there is no gateway buffer overload. Consequently, there is no loss of information packets.

One of the possible options for managing the load of the boundary layer cluster gateway is proposed in [8]. Management is carried out on the basis of a distributed directory system. However, in the case of overload, information packets begin to arrive for processing at the nodes of the fog and cloud layers. Therefore, the processing speed of information packets decreases. Similar problems arise when implementing the boundary layer cluster gateway load management system proposed in [9]. In addition, in the considered works, when conducting research, the possibility of only vertical scaling in the presence of overload is considered.

When implementing the boundary layer cluster gateway load management methodology considered in [10], reinforcement learning is used. But the buffer overflow of the information packet queue is not considered. In [11], when devising a method for virtual clustering of the peripheral environment of the Internet of Things, a load management algorithm is proposed. But this algorithm does not take into account the specificity of GDIoT clusters. For load management, in [12], it is proposed to use the hybrid optimization method "Black Widow" (BWO). This method is focused on energy saving but does not analyze the overload processes.

Queue management methods in multicluster IoT edge architectures are studied in [13]. When gateway buffers are overloaded, these methods redirect the processing of information packets from the edge layers to the cloud layer. This approach significantly increases the processing time of IoT transactions due to delays in transmitting information packets to cloud data centers. In addition, the multicluster architectures under consideration allow for the organization of intercluster exchange, which is unacceptable in GDIoT systems.

The decomposition approach to queue management proposed in [14] is designed for the presence of several autonomous gateways in the cluster. However, the principle of territorial clustering in GDIoT systems assumes the presence of only one gateway in the cluster. The queue management method using artificial neural networks, considered in [7], has limitations on the number of cluster nodes and gateway characteristics. And in the methods proposed in [15], the main criterion is not the processing speed but the load balancing of the cluster nodes. In addition, these methods do not take into account the specific features of the GDIoT architecture.

Therefore, the reviewed scientific papers in the proposed algorithms and load management methods do not take into account the characteristic features of GDIoT systems in detail. In addition, the cited studies do not assume the possibility of horizontal scaling when queue buffers are overloaded. This could lead to both a decrease in the processing speed and to the loss of IoT information packets.

Therefore, it is a relevant task to carry out a study on reducing the number of losses of GDIoT information packets arriving at the boundary layer gateway.

3. The aim and objectives of the study

The aim of our work is to reduce the number of information packet losses in the geographically distributed Internet of Things by devising a method for stabilizing the load control of the boundary layer cluster gateway. For this purpose, it is proposed to use fast temporal horizontal scaling and stabilizing the load control formed in the gateway buffer.

To achieve this goal, the following tasks were set:

– to review possible overload scenarios at the boundary layer of the Internet of Things;

– to develop an algorithm for temporal horizontal scaling;

– to propose a structure for a method for stabilizing the load control over the boundary layer cluster gateway.

4. The study materials and methods

The object of our study is the process of managing overloads at the boundary layer of the geographically distributed Internet of Things. The paper considers GDIoT boundary layer clusters built on the territorial principle. The boundary layer uses GDIoT devices as processing nodes, which include single-board computers with some peripherals. These devices have significantly limited computing resources [16].

The main hypothesis of the study assumes that the implementation of a new method for stabilizing the load control over the boundary layer cluster gateway could reduce the number of losses of Internet of Things information packets. The method is based on temporary horizontal scaling and fixing the thresholds of the gateway buffer. This would ensure an increase in the efficiency of the functioning of the geographically distributed Internet of Things.

The following conditions were used in the development of the method:

Condition 1. GDIoT devices are geographically distributed, and in places of accumulation they have a high density.

Condition 2. Each GDIoT boundary layer cluster will receive information from sensors through a gateway that has a limited buffer for forming a queue of information packets.

Condition 3. Each GDIoT boundary layer cluster has only one gateway.

Condition 4. GDIoT devices used in the boundary layer have limited computing resources.

A number of different methods and algorithms were used in the process of managing the load of the GDIoT boundary layer cluster gateway.

When devising the method for stabilizing the gateway queue, the Incremental Random Early Detection (IRED) algorithm [17] was considered. IRED is an improved packet queue management algorithm in data transmission networks. It is based on the classic Random Early Detection (RED) algorithm, which is used to prevent overload in routing devices.

This algorithm works as follows:

Step 1 – Queue monitoring. The routing device measures the average queue length based on exponential smoothing:

$$q_{avg}(k)=(1-\alpha)q_{avg}(k-1)+\alpha\cdot q, \quad (1)$$

where α is the smoothing coefficient, $0<\alpha<1$; q is the current queue length; k is the current step of the queue analysis cycle.

Formula (1) makes it possible to smooth short-term traffic spikes in order to make decisions based on the long-term state of the queue.

Step 2 is dynamic adjustment of queue thresholds. Unlike RED, in which the thresholds are fixed, in IRED they are adaptively adjusted depending on changes in network traffic:

$$h_{\min}(k)=h_{\min}(k-1)+\beta_{\min}\cdot\left(q_{avg}(k-1)-h_{\min}(k-1)\right), \quad (2)$$

$$h_{\max}(k)=h_{\max}(k-1)+\beta_{\max}\cdot\left(q_{avg}(k-1)-h_{\max}(k-1)\right), \quad (3)$$

where $h_{\min}(k)$, $h_{\max}(k)$ are the current lower and upper queue thresholds; $\beta_{\min}$, $\beta_{\max}$ are the adaptation coefficients.

This action allows the algorithm to dynamically change the boundaries to improve queue management.

Step 3 – calculate the probability of packet rejection. Depending on the average queue length, the algorithm determines whether the incoming packet will be rejected according to the following rules:

– if $q_{avg}(k)<h_{\min}(k)$, then the packet passes without rejection;

– if $h_{\min}(k)\le q_{avg}(k)\le h_{\max}(k)$, then the packet is rejected with a certain probability:

$$p(q_{avg})=p_{\max}\cdot\frac{q_{avg}-h_{\min}}{h_{\max}-h_{\min}}, \quad (4)$$

where $p_{\max}$ is the maximum probability of packet rejection;

– if $q_{avg}(k)>h_{\max}(k)$, then the packet is guaranteed to be dropped.

Step 4 – adaptively adjust the drop probability. The IRED algorithm improves on the conventional RED algorithm by adaptively adjusting the maximum drop probability depending on the network state:

$$p_{\max}(k+1)=p_{\max}(k)+\chi\left(q_{avg}(k)-h_{\min}(k)\right), \quad (5)$$

where χ is the adaptation coefficient.

This allows the algorithm to flexibly respond to changes in traffic intensity, avoiding sharp jumps in the value of the probability of packet loss.

Step 5 is packet rejection or transmission. If the packet is rejected, the sender receives a loss signal (for example, in the TCP protocol this leads to a decrease in the transmission rate). If the packet is not rejected, it is transmitted to the routing device.

Thus, the IRED algorithm takes into account dynamic changes in the network and adaptively adjusts its parameters. This makes it possible to improve queue stability, reduce sharp packet losses, and optimize network throughput.

When selecting an IoT device for temporary horizontal scaling of the boundary layer cluster gateway, the Firefly Algorithm (FA) was used [18]. Each firefly in this algorithm is one of the options for selecting IoT devices for expanding the cluster gateway. For each firefly, the quality of its solution is evaluated, which is considered the brightness of the firefly's radiation.

The algorithm uses the following model of firefly behavior:

– all fireflies can attract each other, regardless of their gender;

– the attractiveness of a firefly for other individuals is proportional to its brightness;

– less attractive fireflies move in the direction of a more attractive firefly;

– the brightness of the emission of a given firefly, visible to other fireflies, decreases with increasing distance between fireflies;

– if a firefly does not see a firefly brighter than itself near it, it moves randomly.

The brightness of the emission of a firefly $s_i\in S$, $i\in 1\,..\,\mathrm{card}\,(S)$ is taken equal to the value of the fitness function $\phi(s_i)$ at its current position. The attractiveness of a firefly s_i for a firefly s_j is taken equal to the value:

$$\beta_{i,j}=\beta_0\cdot\exp\left(-\gamma\cdot r_{i,j}^2\right),\quad i,j\in 1..\mathrm{card}(S), i\neq j, \quad (6)$$

where $r_{i,j}$ is the distance between fireflies s_i, s_j;

β_0 is the mutual attractiveness of fireflies at zero distance between them;

γ is a real value that has the meaning of the light absorption coefficient.

The motion of firefly s_i, which is attracted by a more attractive firefly s_j, is determined from the following formula:

$$X_i = X_j + \beta_{i,j} \cdot \left(X_i - X_j\right) + \alpha U_{|x|}(-1;1),$$
$$i, j \in 1..\mathrm{card}(S), i \neq j, \quad (7)$$

where α is a free randomization parameter; $U_{|x|}(-1;1)$ is a random variable uniformly distributed within (–1,1).

The algorithm scheme looks like this:

1) the initial population of fireflies S is initialized, and the fitness function values are calculated at the initial points;

2) if $\varphi(X_i) < \varphi(X_j)$, then the firefly s_i moves in the direction of the firefly s_j according to formula (7);

3) the fitness function values are calculated at the obtained points;

4) if the condition for the end of the iterations is met, then the best of the current positions of the fireflies is considered an approximate solution to the problem, otherwise the transition to step 2 is carried out.

To assess the effectiveness of the cluster gateway load control method, the probability of losing an information packet was used, which depends on the average gateway load [19]:

$$P_{loss} = P_{loss}\left(\rho_{avg}\right),$$
$$\rho_{avg} = \frac{\lambda_{avg}}{\mu_{avg}}, \quad (8)$$

where λ_{avg} is the average gateway load intensity; μ_{avg} is the average packet service time.

As an additional performance indicator, Q_{perc} was considered – the percentage of the load that arrived at the cluster gateway but was processed on the fog or cloud layers [20].

5. Results of devising and investigating a method for stabilizing load control over the gateway cluster of the edge layer of the Internet of Things

5. 1. Overview of overload scenarios at the edge layer of the Internet of Things

At the lower GDIoT level, two overload scenarios can occur.

The first option is overload at the serving node level. In this case, the packet service rate is less than the packet arrival rate. This leads to a buffer overflow at the serving node. Usually, lower-level GDIoT gateways are overloaded.

Fig. 1 shows an example of such an event. Information packets arriving at the gateway are serviced by component M. If the service component is busy, then the information packets first arrive at the buffer of queue K. Busy buffer elements are marked with the symbol "+". The last free buffer space, marked with the symbol "*", was also busy. Therefore, the next information packet was denied service. Further actions depend on the queue management algorithm to the GDIoT gateway. For example, when using the RED algorithm, a short-term blocking of the incoming flow to the gateway is performed.

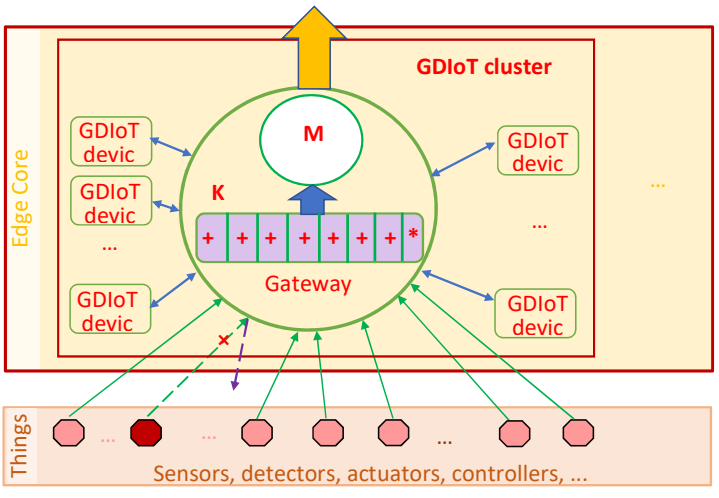

Fig. 1. Gateway overload example

Gateway overload leads to a decrease in the speed of incoming flows and processing time. Overload also results in increased power consumption and packet loss.

The GDIoT edge layer cluster is considered as a queuing system (QS) with one server and N incoming flows. Incoming flows to the cluster gateway are formed by GDIoT sensors. Each sensor can be in two states: active (1) and inactive (0). In the active state, the sensor sends data to the gateway with intensity λ_i. The transition from the inactive state to the active state is carried out with probability p_i, where i is the cluster number. Then the average gateway load intensity is calculated as:

$$\lambda_{averΣ} = \sum_{i=1}^{N} p_i \lambda_i. \quad (9)$$

A generalized queue management scheme for the GDIoT gateway is shown in Fig. 2.

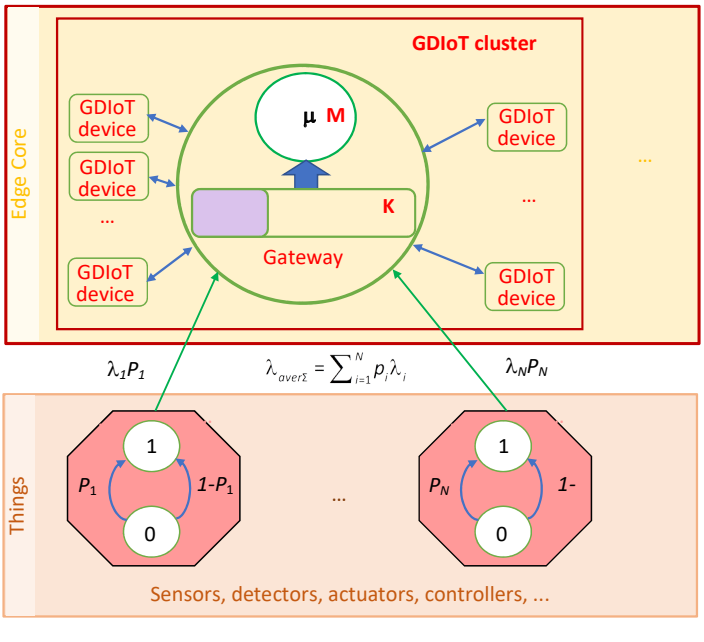

Fig. 2. Generalized queue management scheme for the GDIoT gateway

The second option is channel-level overload. In this case, overload occurs when many active nodes within range simultaneously send data over the communication channel. This option occurs, for example, when using the Carrier Sense Multiple Access (CSMA) protocol.

As a result of such overload, the overall packet delivery rate decreases. Also, some of the side effects of this type of overload are increased energy loss and reduced overall throughput. To counteract channel overload, data link layer mechanisms such as Frequency Division Multiple Access or Time Division Multiple Access are used.

The process of counteracting channel overload is usually divided into several sequential phases. Three phases are considered to reduce the impact of overload on the GDIoT network [21].

The first phase is to detect the load using characteristics such as queue length, channel load, delay, etc.

The second phase is overload notification. When overload is detected, the information is sent to the node to make an appropriate decision. With implicit notification, the overloaded node signals its overload without sending a separate message. This can be an increase in response time, a change in power consumption parameters, an increase in packet loss, etc. This approach allows networks to self-regulate, avoiding critical overload without the need for additional control messages. Explicit notification of overload is carried out by independent control packets, but this approach is rarely used in load control schemes.

The third phase is overload control. In traffic management, control is achieved by adjusting the rate of incoming traffic. In resource management, additional resources are deployed near the overload point. Centralized load control schemes are implemented using a routing protocol with load control. Decentralized load control schemes are distributed and cover all involved devices.

The third phase ends with a decision on overload. In most IoT systems, this solution involves imposing restrictions on the intensity of incoming traffic. But in modern GDIoT networks, the possibility of temporary horizontal scaling has appeared. As an extension of the cluster gateway for the period of overload, one of the IoT devices involved in the boundary layer is selected.

One of the main tasks when making a decision on overload is the need to ensure that there is no loss of information when transmitting it from sensors to the gateway. Typically, in the event of overload, the gateway used resources, that is, vertical scaling was performed. In this case, the speed of processing information coming from IoT devices was significantly reduced. The expansion of the computing capabilities of the boundary layer nodes allowed horizontal scaling in the event of overload.

5. 2. Algorithm of temporary horizontal scaling

Based on the criterion of minimum time of search of the solution when finding a node of a boundary layer for temporary expansion of a gateway, evolutionary algorithms were considered. 6 popular evolutionary algorithms were selected for comparison. The comparison was carried out according to the most important characteristics for the proposed method. Optimization was carried out on test data sets characterizing the state of GDIoT devices based on single-board computers. The relative results of the comparison are given in Table 1.

Table 1

Results of comparing evolutionary algorithms

Algorithm	Speed	Global Optimization	Adaptability to change	Energy consumption
Genetic Algorithm (GA)	Slow	Good	Medium	High
Particle Swarm Optimization (PSO)	High	Easily stuck	High	Low
Fish School Search (FSS)	High	Good	High	Medium
Cuckoo Search (CS)	Medium	Excellent	Low	High
Shuffled Frog Leaping Algorithm (SFLA)	High	Good	High	Low
Firefly Algorithm (FA)	Fastest	Excellent	High	Low

As can be seen from the comparison results, the Firefly Algorithm outperforms other algorithms in terms of the analyzed parameters. This algorithm finds a solution in the shortest time among other algorithms, adapts faster to changes in the load, and avoids local minima. In addition, it does not require a large number of iterations, which has a positive effect on the energy saving of GDIoT devices. Therefore, to implement the Firefly Algorithm, one can use formulas (6) and (7) in step 3 of the stabilizing queue control method.

However, focusing on the specificity of the GDIoT gateway, which connects sensors with the nodes of the boundary layer cluster, one needs to spend as little time as possible on implementing the algorithm. Therefore, to speed up and simplify calculations, the exponential function in formula (6) can be replaced by the Lorentz function. In this case, expression (6) takes the following form:

$$\beta_{i,j}=\frac{\beta_0}{1+\gamma\cdot r_{i,j}^2},\quad i,j\in 1..\mathrm{card}(S), i\neq j. \tag{10}$$

In the Firefly Algorithm, fireflies can move not only to the best solutions but also in random directions. This process is diversification, which is regulated by the randomness parameter α in formula (7)). When intensifying the search, fireflies move to the brightest individuals, implementing a local search. The correct balance between these two processes is critically important. If there is too much intensification, the algorithm may get stuck in a local minimum. If there is too much diversification, the algorithm may go into chaotic mode and not find the optimal solution. Therefore, it is proposed to consistently reduce the free randomization parameter α:

$$\alpha(k)=\alpha_{begin}-(\alpha_{begin}-\alpha_{end})\cdot\exp(-k), \tag{11}$$

where α_{begin}, α_{end} – initial and final values of the randomization parameter; k – number of the current step in the state change cycle.

To accelerate the convergence of the algorithm, another term with a random coefficient α_1 is added to formula (7). Then the movement of the firefly s_i is determined from the following formula:

$$X_i=X_j+\beta_{i,j}\cdot(X_i-X_j)+\alpha U_{|x|}(-1;1)+$$
$$+\alpha_1 U_{|x|}(-1;1)\cdot(X_i-X^*), \tag{12}$$

where X^* is the position of the best firefly for the current epoch; $i, j \in 1..\text{card}(S)$, $i \neq j$.

The proposed algorithm makes it possible to quickly find a boundary layer node that can be used to expand the capabilities of the gateway of the cluster under consideration during the overload period.

5. 3. Structure of the method for stabilizing the load control over the boundary layer cluster gateway

Lower-level GDIoT data transmission protocols are usually supplemented with an active queue management algorithm. This helps achieve optimal network utilization, limited packet loss, and reduced latency [22]. The most popular queue management protocol today is the Random Early Detection (RED) protocol, which helps avoid network overload by preventing mass packet drops. However, in IoT systems with high node density, the traffic flow changes dynamically. At the same time, the conventional implementation of RED requires a complex procedure for making a decision on packet drop. This limits data transmission performance and increases the energy load on the network. Therefore, in IoT systems with high node density, the Incremental Random Early Detection (IRED) algorithm is used for queue management [17]. This algorithm performs adaptive queue management by dynamically changing the thresholds for packet drop depending on the queue fullness.

The main features of the GDIoT system are a significant shift in the data processing process towards decentralized processing and limited energy consumption. Therefore, edge layer clusters are always formed according to the territorial principle. Intercluster convergence of processed data is usually performed on the cloud layer [23]. If there are servers in the fog layer that support several edge clusters, then intercluster convergence can be partially performed on this server. But then the results are still sent to the cloud data center.

The architecture diagram of a separate GDIoT cluster is shown in Fig. 3.

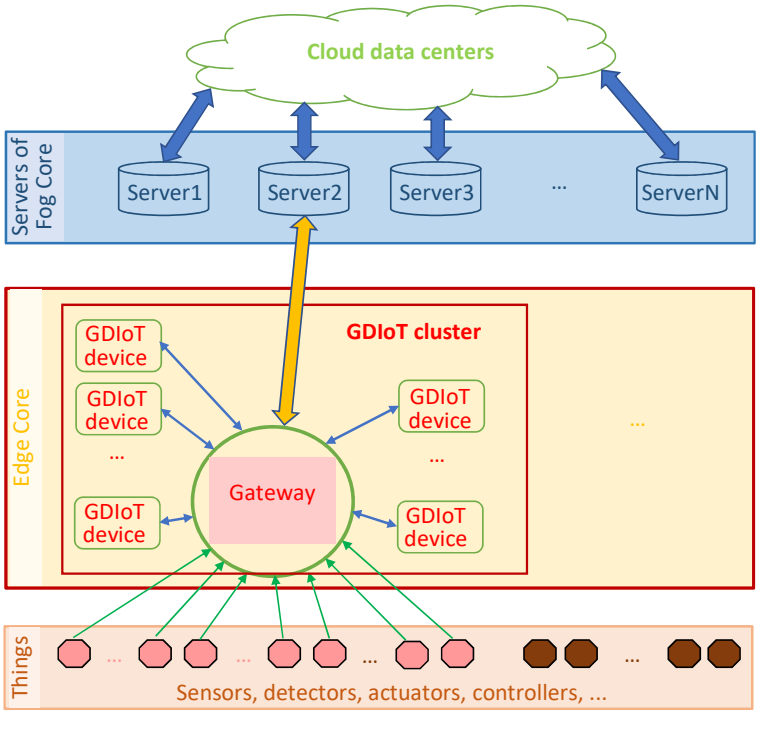

Fig. 3. Architecture diagram of a separate GDIoT cluster

In the GDIoT cluster, the most vulnerable component in terms of overload is the cluster gateway. In the standard scheme using the IRED algorithm, the minimum and maximum thresholds begin to change adaptively as the load increases. This can delay the start of the second phase – sending an overload message. As a result, in the event of a load jump, the gateway may not have enough time to implement the final phase and make a decision. Then some information from the GDIoT sensors will be lost.

To solve this problem with information loss, it is proposed to modify the queue control algorithm for the gateway. Instead of adaptive control, stabilizing control is introduced. In this case, unlike the IRED algorithm, the minimum and maximum thresholds are fixed. The maximum threshold is selected based on the time required to execute the third phase and make a decision on overload.

Below is a step-by-step cycle of the stabilizing queue control process:

Step 1 (preliminary). Determine the lower and upper thresholds of the gateway buffer.

Step 2. When the lower buffer threshold is reached, the procedure for polling GDIoT devices involved in this edge layer cluster is started.

Step 3. When the upper buffer threshold is reached, the procedure for searching for a device for temporary horizontal scaling is started.

Step 4. After determining the device for scaling, the GDIoT data stream is temporarily switched from the gateway to this device.

Step 5. After the queue is reduced to the upper buffer threshold, the gateway resumes receiving the GDIoT data stream, and the temporary expansion node is released.

Step 6. If the queue load decreases to the lower buffer threshold, the procedure for polling GDIoT devices is stopped.

The main criterion in the procedure for finding a device for temporary horizontal scaling is the search time. The device is selected from a set of GDIoT devices involved in the boundary layer and having resources to perform the functions of a temporary gateway.

To assess the effectiveness of the proposed method, a simulation model of the boundary layer cluster of the Internet of Things was used. The cluster served 200 different GDIoT devices. The devices periodically sent information to the gateway, through which information packets were sent for further processing. The gateway queue service buffer was designed for 100 information packets. Changes in the intensity of information flows were formed randomly. 50 GDIoT devices were involved in forming the nodes of the cluster under consideration.

The effectiveness of the proposed stabilizing control (SG) method was evaluated in comparison with standard methods based on the RED and IRED algorithms. The assessment was carried out using the probability of loss of an information packet P_{loss} (expression (8)). Different variants of the average load on the gateway were simulated: $\rho_{avg} \in [0{,}01; 1{,}6]$. The model also analyzed the dependence of indicator Q_{perc} on the average load on the cluster gateway.

The generalized simulation results are shown in the plots of Fig. 4, 5.

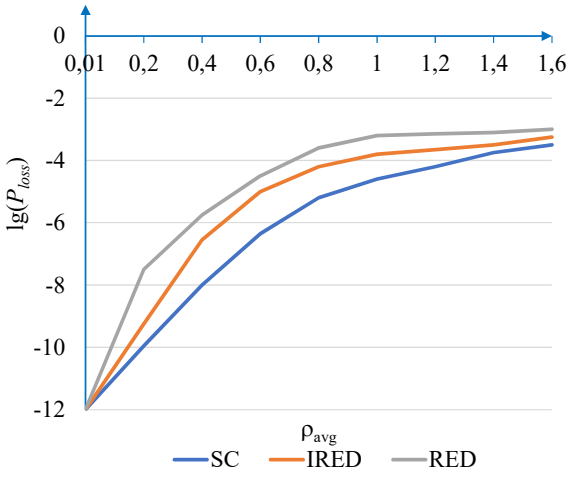

Fig. 4. Dependence of the probability of information packet loss on the load on the gateway

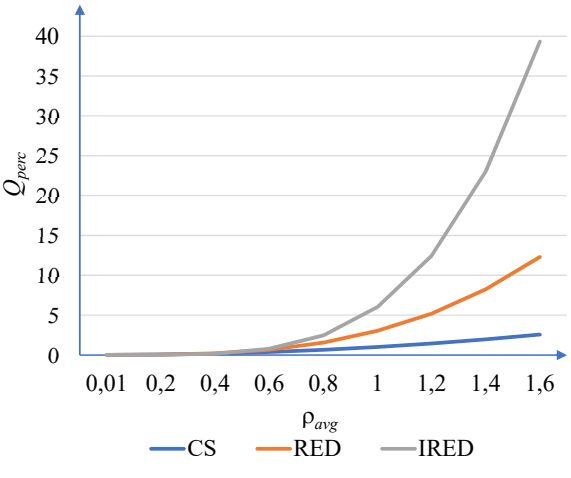

Fig. 5. Dependence of the percentage of information messages processed in fog or cloud layers on the load on the gateway

The average load on the gateway during the simulation varied within the specified limits from minimum load to overload with a step of $\Delta\rho_{avg}=0.2$.

6. Discussion of results based on investigating the method for stabilizing the load control of the boundary layer cluster gateway

We have reviewed possible overload scenarios at the boundary layer of the geographically dispersed Internet of Things. The results allowed us to form a scheme for overloading the cluster gateway (Fig. 1, 2) and prove the possibility of horizontal scaling at the boundary layer. Therefore, during overloading, instead of vertical, horizontal scaling can be performed, which helps accelerate the processing of GDIoT requests. This possibility is explained by taking into account the specific features of the geographically dispersed Internet of Things.

A temporary horizontal scaling algorithm has been developed for the boundary layer cluster gateway of the Internet of Things. The main difference of this algorithm is the speed of finding the required IoT device. The evolutionary Firefly Algorithm (6), (7) was used to implement the algorithm. In order to accelerate the search for the required solution, this algorithm has been modified as follows:

– the exponent in the fitness function (6) is replaced by the Lorentz function (10);

– to accelerate the movement of fireflies to the brightest firefly (formula (7)), the motion formula (11) and (12) have been adjusted.

The modified algorithm allowed us to quickly find a boundary layer node that can be used to expand the capabilities of the gateway of the cluster under consideration during the overload period.

The structure of the method for stabilizing the load control of the gateway of the boundary layer cluster has been proposed. The main difference of this structure from the standard one used in IRED ((1) to (5)) is the fixation of the lower and upper queue thresholds. This allowed us to introduce stabilizing control instead of adaptive control. When modifying the algorithm, the features of the GDIoT architecture were taken into account (Fig. 3). The proposed method allowed us to temporarily horizontally scale the gateway when the upper queue threshold was reached until its buffer was filled. This ensured a reduction in the probability of losing the information package.

Comparative testing of the standard and proposed methods (Fig. 4, 5) showed the following results:

– at a value of ρ_{avg} from 0.2 to 1.2, the use of the proposed method makes it possible to reduce the probability of information packet loss from 10 % to two times;

– the maximum efficiency of the proposed method is achieved at values of ρ_{avg} from 0.5 to 0.9;

– at values of $\rho_{avg}>1.6$, the use of the proposed method is impractical due to overloading of the nodes of the boundary layer;

– at values of $\rho_{avg}>0.8$, the percentage of information messages processed at higher layers begins to decrease significantly compared to standard methods.

The results of our study on the method for stabilizing the gateway load control can be explained by the use of temporary horizontal scaling of the gateway.

Unlike [8], in which the load control method is proposed, the stabilizing control method devised makes it possible not to reduce the rate of packet arrival during overload. This becomes possible due to the timely horizontal scaling of the gateway carried out among the cluster nodes. Unlike [12], in which the queue management method is proposed, the stabilizing control method built has made it possible to reduce the probability of information packet loss. This becomes possible due to the specific features of the GDIoT architecture.

Thus, our results have made it possible to reduce the number of losses of GDIoT information packets arriving at the gateway of the boundary layer cluster. Depending on the average gateway load, the probability of information packet loss decreased from 10 % to two times.

But it is worth noting that the proposed results should be applied with a value of ρ_{avg} from 0.2 to 1.2. In addition, a significant limitation of the study is the presence of only one gateway in the GDIoT boundary layer cluster. The study was also conducted under conditions of high density of GDIoT devices in crowded areas.

A drawback of this study is the absence of restrictions on the number of cluster nodes. To eliminate this drawback, it is necessary to conduct additional studies on the influence of

the number of cluster nodes on the possibility of temporary horizontal scaling.

To advance our study, the following can be noted.

First, it is necessary to conduct a separate study on managing queues of local components of ultra-high density GDIoT sensors. In this case, communication with the boundary layer nodes is carried out using several intelligent gateways. Each gateway provides communication with a separate boundary layer cluster. Secondly, it is necessary to consider the possibility of joint management of queues of information packets to the gateways of this local component.

7. Conclusions

1. We have reviewed possible overload scenarios at the edge layer of the geographically distributed Internet of Things. Three phases were identified that are used to reduce the impact of overload on the GDIoT network. Special attention was paid to the overload scenario at the level of the serving node of the GDIoT sensor group. The specific features of the GDIoT edge layer were taken into account during development. This allowed us to propose the use of horizontal scaling when the gateway queue is overloaded.

2. A temporal horizontal scaling algorithm has been developed for the gateway of the Internet of Things edge layer cluster. The main difference of this algorithm from existing ones is the speed of finding the required IoT device. To this end, the evolutionary Firefly Algorithm was used. In order to speed up the search for the required solution in this algorithm, the exponent in the fitness function was replaced by the Lorentz function. In addition, the motion formula was adjusted to accelerate the movement of fireflies to the brightest firefly. This algorithm allowed us to devise a method for stabilizing queue control at its overload.

3. The structure of the method for stabilizing load control of the boundary layer cluster gateway has been proposed. The main difference of this structure from the standard one used in IRED ((1) to (5)) is the fixation of the lower and upper queue thresholds. This modification has made it possible to introduce stabilizing control instead of adaptive control and to perform temporary horizontal scaling of the gateway until its buffer is filled. This ensured a reduction in the number of losses of GDIoT information packets arriving at the boundary layer gateway. The results of the study have made it possible to compare the effectiveness of the standard and proposed methods according to the criterion of the probability of loss of an information packet. At a small average load and when the threshold of 1.6 is exceeded, the proposed method has no advantage in terms of the probability of packet loss over the standard ones. At an average load on the gateway from 0.2 to 1.2, the proposed method makes it possible to reduce the probability of loss of an information packet from 10 % to two times. It has also been proven that the maximum efficiency of the proposed method is achieved at average load values from 0.5 to 0.9.

Conflicts of interest

The authors declare that they have no conflicts of interest in relation to the current study, including financial, personal, authorship, or any other, that could affect the study, as well as the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available, either in numerical or graphical form, in the main text of the manuscript.

Use of artificial intelligence

The authors confirm that they did not use artificial intelligence technologies when creating the current work.

References

1. Gamboa, A., Villazón, A., Meneses, A., Ormachea, O., Orellana, R. (2024). Altitude's Impact on Photovoltaic Efficiency: An IoT-Enabled Geographically Distributed Remote Laboratory. Smart Technologies for a Sustainable Future, 133–144. https://doi.org/10.1007/978-3-031-61905-2_14
2. Singh, S. P., Kumar, N., Kumar, G., Balusamy, B., Bashir, A. K., Dabel, M. M. A. (2025). Enhancing Quality of Service in IoT-WSN through Edge-Enabled Multi-Objective Optimization. IEEE Transactions on Consumer Electronics, 1–1. https://doi.org/10.1109/tce.2025.3526992
3. Yan, M. (2024). Receive wireless sensor data through IoT gateway using web client based on border gateway protocol. Heliyon, 10 (11), e31625. https://doi.org/10.1016/j.heliyon.2024.e31625
4. Kuchuk, N., Kashkevich, S., Radchenko, V., Andrusenko, Y., Kuchuk, H. (2024). Applying edge computing in the execution IoT operative transactions. Advanced Information Systems, 8 (4), 49–59. https://doi.org/10.20998/2522-9052.2024.4.07
5. Kuchuk, H., & Malokhvii, E. (2024). Integration of IoT with cloud, fog, and edge computing: a review. Advanced Information Systems, 8 (2), 65–78. https://doi.org/10.20998/2522-9052.2024.2.08
6. Alwakeel, A. M. (2021). An Overview of Fog Computing and Edge Computing Security and Privacy Issues. Sensors, 21 (24), 8226. https://doi.org/10.3390/s21248226
7. Naveen, S., Kounte, M. R., Ahmed, M. R. (2021). Low Latency Deep Learning Inference Model for Distributed Intelligent IoT Edge Clusters. IEEE Access, 9, 160607–160621. https://doi.org/10.1109/access.2021.3131396
8. Hao, L., Naik, V., Schulzrinne, H. (2022). DBAC: Directory-Based Access Control for Geographically Distributed IoT Systems. IEEE INFOCOM 2022 - IEEE Conference on Computer Communications, 360–369. https://doi.org/10.1109/infocom48880.2022.9796804

9. Samir, A., Dagenborg, H. (2023). Adaptive Controller to Identify Misconfigurations and Optimize the Performance of Kubernetes Clusters and IoT Edge Devices. Service-Oriented and Cloud Computing, 170–187. https://doi.org/10.1007/978-3-031-46235-1_11
10. Cui, H., Tang, Z., Lou, J., Jia, W. (2023). Online Container Scheduling for Low-Latency IoT Services in Edge Cluster Upgrade: A Reinforcement Learning Approach. 2023 IEEE/CIC International Conference on Communications in China (ICCC), 1–6. https://doi.org/10.1109/iccc57788.2023.10233668
11. Kuchuk, H., Mozhaiev, O., Kuchuk, N., Tiulieniev, S., Mozhaiev, M., Gnusov, Y. et al. (2024). Devising a method for the virtual clustering of the Internet of Things edge environment. Eastern-European Journal of Enterprise Technologies, 1 (9 (127)), 60–71. https://doi.org/10.15587/1729-4061.2024.298431
12. Vaiyapuri, T., Parvathy, V. S., Manikandan, V., Krishnaraj, N., Gupta, D., Shankar, K. (2021). A Novel Hybrid Optimization for Cluster-Based Routing Protocol in Information-Centric Wireless Sensor Networks for IoT Based Mobile Edge Computing. Wireless Personal Communications, 127 (1), 39–62. https://doi.org/10.1007/s11277-021-08088-w
13. Azimi, S., Pahl, C., Shirvani, M. (2020). Particle Swarm Optimization for Performance Management in Multi-cluster IoT Edge Architectures. Proceedings of the 10th International Conference on Cloud Computing and Services Science. https://doi.org/10.5220/0009391203280337
14. Kuchuk, H., Kalinin, Y., Dotsenko, N., Chumachenko, I., Pakhomov, Y. (2024). Decomposition of integrated high-density IoT data flow. Advanced Information Systems, 8 (3), 77–84. https://doi.org/10.20998/2522-9052.2024.3.09
15. Kuch Kuchuk, H., Husieva, Y., Novoselov, S., Lysytsia, D., Krykhovetskyi, H. (2025). Load balancing of the layers IoT fog-cloud support network. Advanced Information Systems, 9 (1), 91–98. https://doi.org/10.20998/2522-9052.2025.1.11
16. Hunko, M., Tkachov, V., Kovalenko, A., Kuchuk, H. (2023). Advantages of Fog Computing: A Comparative Analysis with Cloud Computing for Enhanced Edge Computing Capabilities. 2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek). https://doi.org/10.1109/khpiweek61412.2023.10312948
17. Simaiya, S., Shrivastava, A., Keer, N. P. (2014). IRED Algorithm for Improvement in Performance of Mobile Ad Hoc Networks. 2014 Fourth International Conference on Communication Systems and Network Technologies, 283–287. https://doi.org/10.1109/csnt.2014.62
18. Qasim, M., Sajid, M. (2024). An efficient IoT task scheduling algorithm in cloud environment using modified Firefly algorithm. International Journal of Information Technology, 17 (1), 179–188. https://doi.org/10.1007/s41870-024-01758-5
19. Li, J., Zhou, T. (2024). Data-Driven Fully Distributed Load Frequency Control for an IoT-Based Interconnected Grid Considering a Performance-Based Frequency Regulation Market. IEEE Internet of Things Journal, 11 (17), 28692–28704. https://doi.org/10.1109/jiot.2024.3402274
20. Petrovska, I., Kuchuk, H., Kuchuk, N., Mozhaiev, O., Pochebut, M., Onishchenko, Y. (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), 1–6. https://doi.org/10.1109/dessert61349.2023.10416496
21. Carvalho, D., Sullivan, D., Almeida, R., Caminha, C. (2022). A Machine Learning Approach to Solve the Network Overload Problem Caused by IoT Devices Spatially Tracked Indoors. Journal of Sensor and Actuator Networks, 11 (2), 29. https://doi.org/10.3390/jsan11020029
22. Sobchuk, V., Pykhnivskyi, R., Barabash, O., Korotin, S., Omarov, S. (2024). Sequential intrusion detection system for zero-trust cyber defense of IOT/IIOT networks. Advanced Information Systems, 8 (3), 92–99. https://doi.org/10.20998/2522-9052.2024.3.11
23. Sundaram Paulraj, S. S., Kannabiran, V. (2024). Neuro-fuzzy-based cluster formation scheme for energy-efficient data routing in IOT-enabled WSN. International Journal of Communication Systems, 38 (3). https://doi.org/10.1002/dac.5984