INFORMATION TECHNOLOGY

The object of this research is the process of load balancing in distributed Internet of Things (IoT) systems. Within this work, a complex of problems related to efficient load distribution has been addressed. The authors conducted an analysis of existing load-balancing approaches and their drawbacks and proposed an enhanced architecture for the MQTT broker. Additionally, methods and algorithms for load balancing were developed based on multi-criteria server monitoring.

Furthermore, the authors created a mathematical model to assess the uniformity of load distribution in the system and introduced a corresponding metric – the load distribution coefficient. In order to evaluate the proposed load balancing methods, a series of experiments were conducted, including the simulation of a distributed IoT system with non-deterministic load. The main goal of these experiments was to assess the uniformity of MQTT load distribution by the broker.

The results of the experiments confirmed the hypothesis of improved load distribution efficiency through multi-criteria monitoring-based balancing. The utilization of the proposed load-balancing methods allowed for a more efficient utilization of computational resources. It was found that when using the proposed methods, in the case of non-deterministic load in the IoT system, the load distribution coefficient on average exceeded the corresponding indicator of existing methods by 70 %. In addition, the value of this coefficient for the proposed methods remains virtually unchanged throughout the experiment, which is evidence of the stable operation of the system as a whole. The results obtained can be useful in the development of modern IoT systems

Keywords: internet of things, load balancing, cloud computing, distributed systems, performance evaluation

UDC 621.382.2

DOI: 10.15587/1729-4061.2023.287790

IMPROVING A PROCEDURE OF LOAD BALANCING IN DISTRIBUTED IOT SYSTEMS

Ihor Zakutynskyi
Postgraduate Student
Department of Electronics, Robotics, Monitoring and IoT Technologies**
Ihor Rabodzei*
Corresponding author
E-mail: igor.rabodzei@gmail.com
Stanislav Burmakin
Postgraduate Student
Department of Computer Information Technologies**
E-mail: svburmakin@gmail.com
Oleksandr Kalishuk*
Vitalii Nebylytsia*
*Department of Information Technology Security**
National Aviation University
Liubomyra Huzara ave., 1, Kyiv, Ukraine, 03058

Received date 21.07.2023
Accepted date 28.09.2023
Published date 30.10.2023

***How to Cite**: Zakutynskyi, I., Rabodzei, I., Burmakin, S., Kalishuk, O., Nebylytsia, V. (2023). Improving a procedure of load balancing in distributed iot systems. Eastern-European Journal of Enterprise Technologies, 5 (2 (125)), 6–22. doi: https://doi.org/10.15587/1729-4061.2023.287790*

1. Introduction

The Internet of Things concept is one of the most promising technologies of the 21st century. The number of connected IoT devices is growing exponentially, and according to the IoT Analytics report [1], it will reach 27 billion by 2025. This growth is due to the development of the base of electronic components, the transition to new generations of wireless communication, as well as a decrease in the cost of electronics production in general. Also, an important factor in the development of IoT technologies is the growth of the power of computing resources and the development of cloud technologies. Since the number of connected devices is constantly growing, the amount of data they generate is also constantly increasing. This trend generates high requirements for data transmission, storage, and processing systems.

In modern Internet of Things systems, the main data transfer protocol between connected devices and servers is MQTT (Message Queuing Telemetry Transport). MQTT can work under conditions of loss of communication, and it also creates a small load on the data transfer channel. This allows for fast information exchange between devices and the server, which is critical for an IoT network. This protocol uses the Puslisher-Subscriber architecture. According to this architecture, the routing of messages between system components is performed by the MQTT broker. Under the standard mode of operation, the broker routes incoming messages to all clients who are subscribed to a certain "topic". However, with the growth of the number of connected devices and the amount of data generated, there is a need to scale the system and balance the load in it.

To ensure the robustness and scalability of the IoT system, distributed dynamic computing methods are used. Such methods make it possible to perform time-consuming computing tasks on several servers connected to one network. In the context of IoT systems based on the MQTT protocol, load distribution is usually performed by introducing additional modules, such as load balancers or queuing systems. With this approach, one more software component is needed, which will be an "intermediary" between the MQTT broker and the servers on which the actual data processing takes place. The use of this approach has the following drawbacks and potential problems:

– increase in delays: the introduction of an additional "intermediary" component between the MQTT broker and computing servers increases delays in data transmission and processing;

– complexity of configuration and management: additional software components complicate the process of system deployment and maintenance;

– the need for additional resources: additional components require additional computing resources, which increases the cost and complexity of the system as a whole.

A potentially more optimal approach is to implement load balancing at the MQTT broker level. In this case, the system does not require additional components, which leads to the reduction of message delivery time between the client (IoT device) and the subscriber (computing server) to a minimum. In addition, the implementation of this approach allows existing IoT systems to use distributed computing without changing the system architecture.

Optimizing the use of computing resources helps improve system performance and reliability, and reduces hardware and support costs. Therefore, the task of load balancing at the MQTT broker level has the potential to optimize the operation of the IoT system as a whole and requires further research and development of new solutions.

2. Literature review and problem statement

One of the most popular research problems is load balancing in systems based on cloud and fog computing. For example, in [2], the authors propose a method for balancing the input load between virtual machines, based on the change in CPU load. However, this approach has limitations since the input load may be oriented not on processor resources but on memory or network resources. Thus, this approach to balancing will not be effective. In [3], the authors also use the concept of virtual machines in cloud systems. However, the effectiveness of the proposed load distribution between them has not been fully investigated. The authors of study [4] proposed a method of load distribution based on «slicing» of network resources in the context of fog computing. However, the concept of fuzzy computing, and therefore the proposed methods, cannot always be applied in an IoT network.

Another approach to solving load distribution tasks is balancing based on geocoordinates. In study [5], the authors propose a load distribution model based on the geographic location of connected servers (Geography-Aware). In the context of the Internet of Things system, these methods are limited because the connected devices are often not geographically distributed. Also, a study of load distribution based on these algorithms was carried out in [6]. In the cited study, the authors propose approaches for load balancing in data storage systems.

Algorithms based on geodistribution are also used in [7] to solve the problem of distributed data centers. Such a model is efficient for the above systems but not efficient enough for systems with dynamic and hard-to-predict load, such as IoT networks.

In [8], the authors proposed a complex load distribution system in the IoT network. The concept of that system is based on the dynamic allocation of resources for each part of the network and the segmentation process. The disadvantages of the system include the complexity of implementation due to the need to provide an additional level of fuzzy calculations.

In [9], a comprehensive analysis of load balancing methods in cloud and fog computing, as well as the possibility of their application in IoT systems, is carried out. However, no scientific and practical experiments were conducted to evaluate the effectiveness of the considered methods in the context of IoT systems, which is a limitation of the cited study. In [10], the authors consider energy-efficient load balancing methods in fog computing systems. This approach is interesting and promising as it focuses on the overall energy efficiency of the system but, at the same time, it is limited due to the need to implement an additional level of fuzzy computing. A similar study was carried out in [11], where the application of existing balancing methods in fog computing systems was considered. The results reported in the cited study are also limited in the context of Internet of Things systems, due to the need for an additional computing layer.

Another approach to solving the problem of load balancing is the use of machine learning methods. The authors of [12] propose a load balancing scheme based on neural networks for Internet of Things systems. To determine the effectiveness of this method, the authors performed mathematical modeling, but the operation of these methods in the context of dynamic systems with a distributed architecture was not investigated. A similar method is proposed in [13], where the topology of the network and ways of distributing the load in it were considered in detail. The disadvantages of the study include the complexity of implementation and the complexity of the system architecture. The authors of study [14] propose load balancing methods for multipath routing. However, in the cited study there are no practical experiments that would confirm or refute the effectiveness of the above methods.

Another area of research into the problem of load balancing is the application of mathematical modeling methods. The authors of [15] proposed a mathematical theory of dynamic load balancing in cellular networks. Among the shortcomings of this approach, we can single out the limited adaptability of the theory to changes in the intensity of the input load of the network. In [16], the authors consider a quasi-positional algorithm for load balancing in a cloud computing environment. Mathematical models for determining the optimal number of computing containers in Internet of Things systems are proposed in [17]. The main drawback of this approach may be the difficulty of adapting models to changes in system configurations and loads. Study [18] considers an optimization model for scheduling tasks in cloud computing but this model does not take into account dynamic load changes. The authors of work [19] propose a mathematical model for increasing the efficiency of using database resources in cloud computing.

In addition, many studies consider methods for improving the efficiency of load distribution in industrial systems. For example, work [20] investigates the problems of load balancing on the example of a implemented system for monitoring and managing public transport. However, those studies do not fully reveal the problem of load balancing in distributed IoT systems as they only consider available commercial solutions.

The main limitation of the above studies is their focus on classic web applications and commercial solutions; they do not sufficiently take into account the dynamic load that occurs in Internet of Things systems.

Therefore, there is a need to devise an effective procedure for load balancing in distributed IoT systems. This procedure should ensure an even distribution of tasks between available workers and the most efficient utilization of server resources.

3. The aim and objectives of the study

The aim of this study is to improve the procedure of load balancing in distributed systems of the Internet of Things

based on multi-parameter monitoring. This will make it possible to improve the uniformity of load distribution in dynamic systems of the Internet of Things, and therefore to increase the efficiency of the use of computing resources.

To achieve the goal, the following tasks were set:

– to develop an improved MQTT broker architecture with support for dynamic load balancing methods;

– to build a mathematical model for determining the load factor of the computing server;

– to develop a dynamic load balancing algorithm;

– to construct a mathematical model for assessing the uniformity of load distribution in distributed systems of the Internet of Things;

– to conduct experimental studies to determine the effectiveness of the proposed methods.

4. The study materials and methods

4. 1. The object and hypothesis of the study

The object of research is the process of load balancing in distributed systems of the Internet of Things, in particular at the level of the MQTT protocol.

Under the standard mode of operation of MQTT (Fig. 1), a client subscribed to a certain topic has access to a copy of each message broadcast to this topic [21]. In this case, to balance the load, it is necessary to distribute it by topics, and to perform the subscription of workers to a specific topic.

But this approach has certain disadvantages because:

1. The load in IoT systems is non-deterministic and difficult to predict, so the early distribution of the load between workers is impossible in real systems.

2. The workload of workers will be uneven.

3. Load distribution using partitions is incorrect within the concept of the MQTT protocol.

This is especially true for systems that generate large volumes of data (BigData).

A possible solution is to use the "Shared Subscriptions" method, which appeared in the MQTT 5 release [22]. With shared subscriptions (Fig. 2), clients who share a subscription in the same group receive messages one at a time – a process sometimes called client load balancing. The message load of one topic is distributed among all subscribers (Fig. 2). When using the Shared Subscriptions method, the MQTT broker performs sequential routing of messages between available servers of a certain group. With this approach, each worker receives 1/N messages, where N is the number of servers in the group.

But this approach also has disadvantages related to the uneven distribution of the load. Messages received at different points in time, depending on their type, may require different software processing and, therefore, generate different loads. Thus, there will be situations where part of the servers will be fully loaded and will not be able to perform some tasks (or perform them with a delay), while the other part will be inactive or under-loaded at the same time.

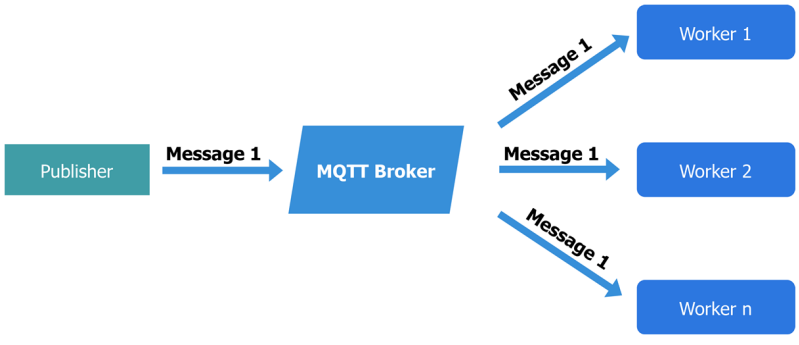


Fig. 1. MQTT standard subscription method

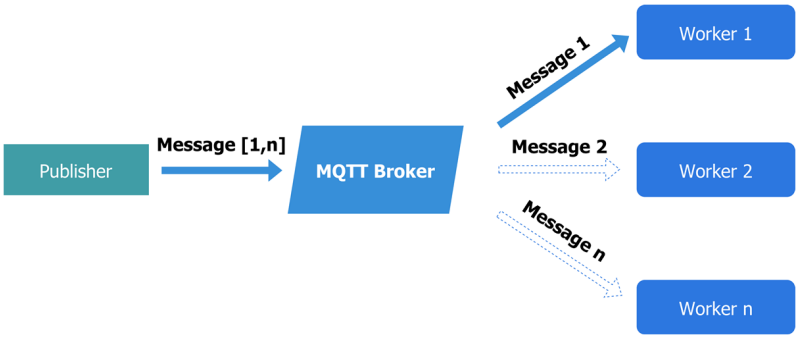


Fig. 2. MQTT shared subscriptions mechanism

The research hypothesis assumes that load balancing based on multi-parameter monitoring can improve the efficiency of computing resources in Internet of Things systems with a distributed architecture.

Two MQTT brokers with different balancing methods were used for the our experiment. Namely, HiveMQ with the Shared Subscriptions method (hereinafter Method 1) as well as the broker was developed based on the methodology proposed in this study (hereinafter Method 2).

4. 2. Research methodology

In the course of this study, a methodology (Fig. 3) based on mathematical modeling and experimental methods was used.

At the first stage, an analysis of available methods of load balancing in Internet of Things systems was carried out. In addition, their limitations and shortcomings in the context of distributed systems with dynamic load were determined. Based on the analysis, a goal was formulated and a hypothesis was put forward regarding the improvement of existing load balancing methods, as well as the research task was stated. Next, the MQTT broker architecture was developed, with monitoring and load balancing modules.

At the next stage, a mathematical model was proposed for determining the load factor of the computing server. Also, based on the proposed model, a balancing algorithm was developed, and the software implementation of the developed algorithm in the load balancing module was also performed.

To evaluate the effectiveness of the proposed methods, a mathematical model for assessing the uniformity of load distribution in distributed systems of the Internet of Things has been built. This model is constructed on the basis of vector distances of instantaneous load of active computing servers. According to the proposed mathematical model, the appropriate coefficient was introduced and a software environment was developed for its determination.

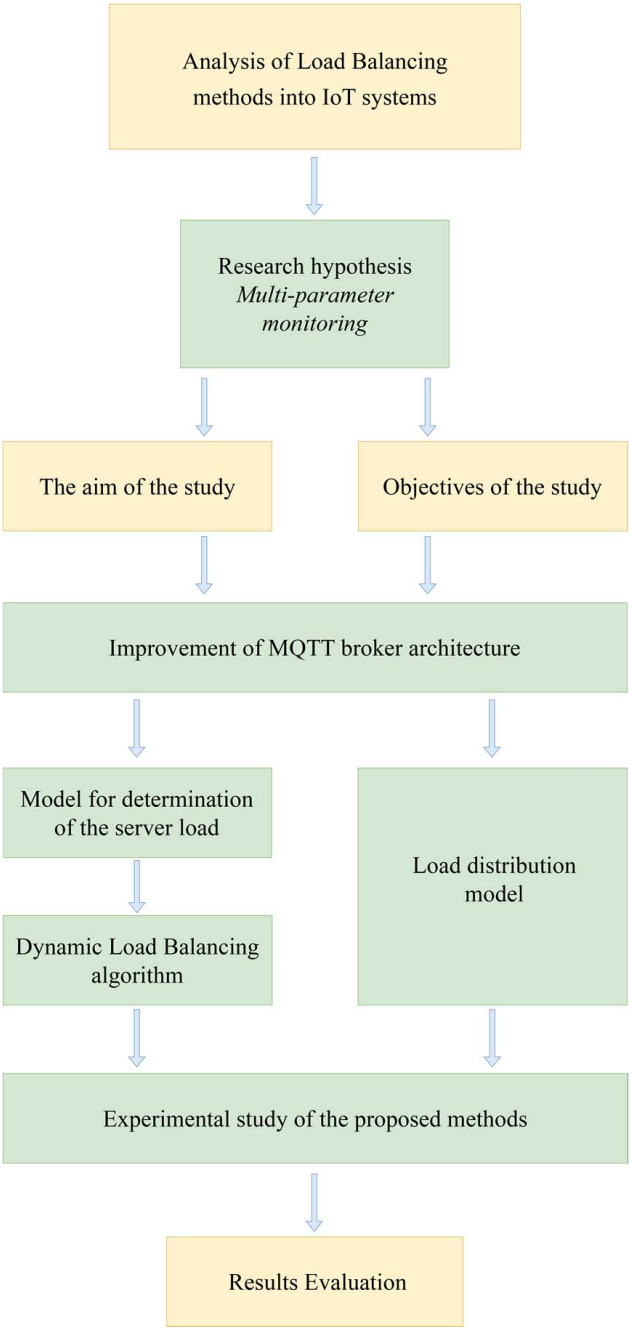


Fig. 3. Methodology of the study

At the final stage, a number of practical experiments were conducted, in which the impact of existing and proposed balancing methods and the efficiency of using server resources by the Internet of Things system with dynamic load were investigated.

5. Results of investigating the improved load balancing method in distributed IoT systems

5. 1. MQTT broker

5. 1. 1. Architecture

For optimal load distribution between available servers (workers), the MQTT broker must receive information about the current load status of each available server, and route the message to the least loaded one at the moment. Fig. 4 shows an improved model of the MQTT protocol. In this model, two additional modules are introduced in the MQTT broker – Utilization monitoring and Load Balancer. In the proposed version, an additional channel is introduced for the monitoring system – Monitoring channel (TCP, 8123 Port). Through the above MQTT channel, the broker receives information about the load of each server. The evaluation is based on the following characteristics: CPU, RAM, Disk, and Network usage. After receiving the monitoring package, the broker stores them in a special structure – Utilization state. This structure, in turn, is used by the load balancing module (Load balancer).

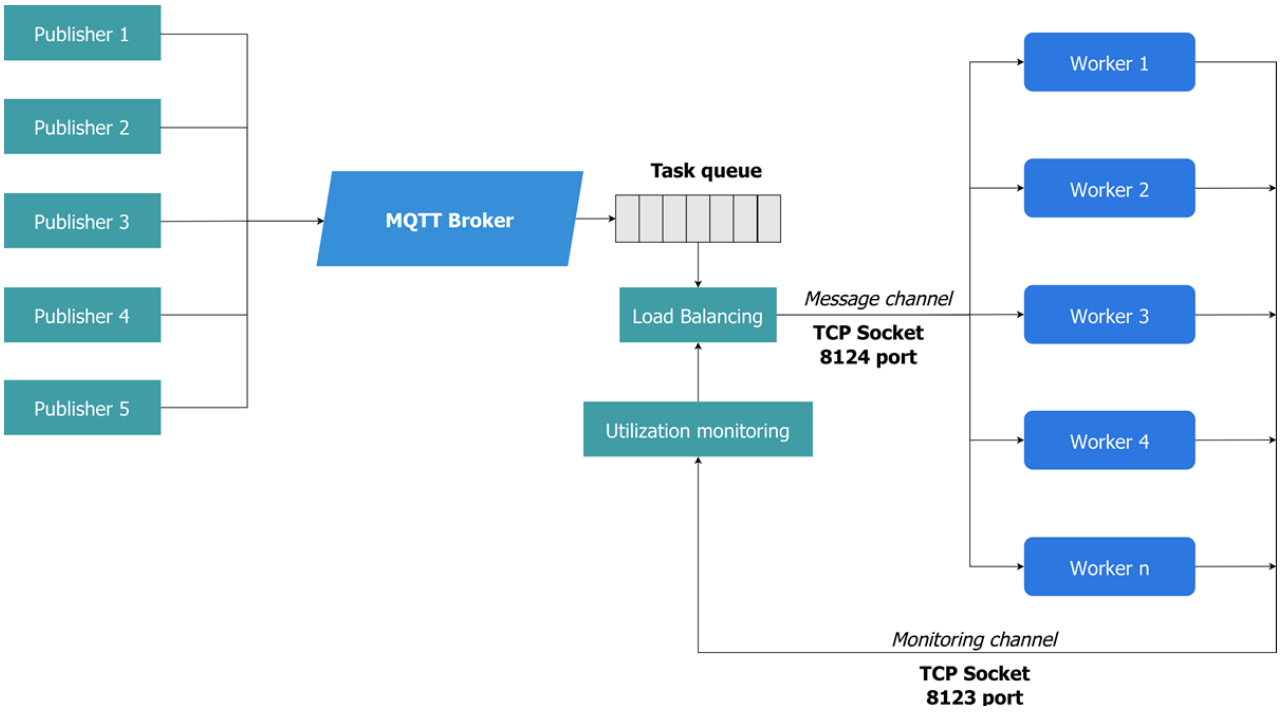


Fig. 4. Improved MQTT protocol model

Load balancer is an asynchronous module that routes incoming messages from Publishers to Subscribers (workers) over a standard MQTT channel (TCP, 8124 Port).

5. 1. 2. Monitoring module

The monitoring module determines and analyzes instantaneous load values for each indicator. The above module consists of three main components:

– server monitoring service – determines the current value of the server load, and transmits it via the monitoring transmission channel. In the proposed implementation, it is part of the MQTT client library;

– transmission channel – TCP socket through which monitoring data is transmitted. In this case, serialized JSON is transmitted (Table 1);

– monitoring information processing and analysis service – the MQTT component of the broker, which receives, stores, and analyzes the load values for each connected server.

Table 1

Structure of the object with monitoring data

Field	Type
utilization	Number
cpu_utilization	Number
ram_utilization	Number
disk_utilization	Number
network_utilization	Number

5. 2. Mathematical model for determining the computing server load factor

For load balancing, a general indicator (1) of server load, Load Score, is introduced, which is determined on the basis of the received monitoring data.

$$LoadScore(w) = \\ = \int_0^T \begin{bmatrix} W_{CPU} CPU(w,t) + W_{RAM} RAM(w,t) + \\ + W_{DISK} DISK(w,t) + W_{NET} NET(w,t) \end{bmatrix} dt, \quad (1)$$

where w_{CPU}, w_{RAM}, w_{DISK}, w_{NET} are weighting factors for the characteristics of CPU, RAM, Disk utilization, and Network utilization, respectively.

$CPU(w, t)$, $RAM(w, t)$, $Disk(w, t)$, $NET(w, t)$ reproduce the functions of resource use by server w at time t.

In this case, additional coefficients are introduced for each characteristic, which makes it possible to dynamically change the influence of one or another parameter on the load distribution in the system. The use of dynamic coefficients makes it possible to avoid overloading specific resources. For example, if a server has a large amount of RAM and at the same time is limited by CPU resources, it is possible to reduce the ratio of wRAM and increase wCPU, which will increase the overall performance of the server.

5. 3. Dynamic load balancing algorithm

In the process of load balancing for active servers, an additional variable, Threshold, is introduced. The Threshold value is set based on the permissible load that the server can withstand without being overloaded. When a server's LoadScore exceeds this threshold, it means that the server's resource usage (CPU, RAM, disk, and network usage) has reached a point where it is considered too high for optimal performance.

Fig. 5 shows the dynamic load distribution algorithm in the system.

Servers with LoadScores below the Threshold are considered to have a manageable load and can receive new messages (tasks). Servers with LoadScores above the Threshold should not receive messages to prevent further overload.

Algorithm 1 Load Balancing Algorithm

Require: Set of workers: $W = \{w_1, w_2, ..., w_n\}$
Require: Metrics for each worker w:
$CPU(w)$ {CPU utilization}
$RAM(w)$ {RAM usage}
$Disk(w)$ {Disk utilization}
Require: Load weights: CPU_{weight}, RAM_{weight}, $Disk_{weight}$
for each worker w in W **do**
  CollectData(w, Δt) {Collect CPU, RAM, Disk data}
  Calculate $LoadScore(w)$ {Based on load weights}
end for
Overload threshold: $Threshold$
Set of available servers below threshold:
$AvailableServers = \{w \in W | LoadScore(w) < Threshold\}$
Ranked workers list: $R = [w_1, w_2, ..., w_n]$ sorted by $LoadScore(w)$ in ascending order
Received new message with load preferences: $Message = (CPU_{pref}, RAM_{pref}, Disk_{pref})$
for each worker s in ranked list R **do**
  if $(CPU(w) \leq CPU_{pref})$ and $(RAM(w) \leq RAM_{pref})$ and $(Disk(w) \leq Disk_{pref})$ **then**
    Send $Message$ to worker w and break
  end if
end for
Continuously update load scores based on real-time data

Fig. 5. Dynamic load balancing algorithm

5. 4. Mathematical model for assessing the uniformity of load distribution in distributed Internet of Things systems

To evaluate the effectiveness of balancing, we shall introduce a special characteristic – the coefficient of uniformity of load distribution. To this end, a combination of statistical metrics for each parameter that affects server load, as well as an aggregated metric for the entire system, is applied. One possible approach is to use "distances" between servers based on the values of the load parameters. Euclidean differences between parameter vectors are used to calculate "distances". In our case, the vectors of instantaneous server load values are x_1, x_2, x_i, where x_i is the vector of measurements (2) for the i-th server:

$$x_i = (CPU_i, RAM_i, DISK_i, NET_i). \quad (2)$$

Then the Euclidean distance between two vectors x_i and x_j can be calculated by the formula:

$$\text{Distance}(x_i, x_j) = \sqrt{\sum_{k=1}^{4} (x_{ik} - x_{jk})^2}, \quad (3)$$

where x_{ik} is the k-th coordinate of the vector x_i, x_{jk} is the k-th coordinate of the vector x_j.

Thus, the Euclidean distance between each pair of servers can be calculated. In the next step, one can use these distances to calculate an aggregated average distance metric:

$$\text{Average Distance} = \frac{1}{N(N-1)} \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} \text{Distance}(x_i, x_j). \quad (4)$$

This metric will show the uniformity of load distribution between servers. If the value of this metric is large, it may indicate an uneven distribution of the load, and if it is small, it may indicate an even distribution.

5. 5. Experimental study of the effectiveness of the proposed methods

5. 5. 1. Experiment environment

To emulate the non-uniform load generated in real systems, 50 tasks of different types and intensities were generated. Table 2 gives an example of experimental tasks. Based on these MQTT data, clients form messages and send them to the MQTT broker, which in turn routes it to the corresponding server (worker) according to a certain algorithm.

After the worker receives the message, the program code is executed, which generates the load according to the received coefficients: CPU/RAM/Disk/Network – intensive.

In this experiment, 5 servers were used as workers, the characteristics of which are given in Table 3.

The experiment involves the use of servers with identical characteristics and on the same local network, to effectively evaluate balancing algorithms.

Table 2

Experimental tasks

Task number	Complexity of the task			
	CPU	RAM	Disk I/O	Network
1	50	46	21	32
2	12	78	12	24
3	1	32	55	21
4	16	76	6	75
5	9	11	22	87
6	44	7	37	14
7	1	9	61	53
8	12	43	29	32
9	43	30	18	98
10	4	12	14	16

Table 3

Computation server parameters

Indicator	Value
Machine Type	Basic
CPU Type	Regular Intel
Cores	1
RAM	512Mb
Disk Type	SSD
Memory	8Gb
Network Bandwidth	1Gb/s
Operating System	Debian

5. 5. 2. Evaluation of results

The main characteristic when comparing balancing methods is the proposed coefficient of uniformity of load distribution. To calculate this coefficient, it is necessary to store the instantaneous load values of each server at a certain interval. In the case of the proposed architecture, this can be done at the broker level since it receives load indicators from workers in real time. But the HiveMQ broker does not receive this information, so for the experiment it is necessary to transfer monitoring to a separate module – Load Monitoring.

In this experiment, the monitoring server receives load indicators from workers with an interval of 100 ms, and calculates the coefficient of uniformity of load distribution for each set of received values. Fig. 6 shows the general scheme of the experiment.

Execution of tasks: charts in Fig. 7, 8 show the execution time and the distribution of tasks among 5 servers.

The above charts make it possible to visually assess the difference in the distribution of tasks among servers for the two investigated methods (Method 1 and Method 2) at different points in time. Based on this data, it is possible to identify the main load peaks or server overloads.

Table 4 gives the result of balancing by the algorithm based on Method 1, and Table 5 – based on Method 2, for each of the five experimental servers.

Fig. 9–14 show the dynamics of changes in the main server load characteristics for each balancing method.

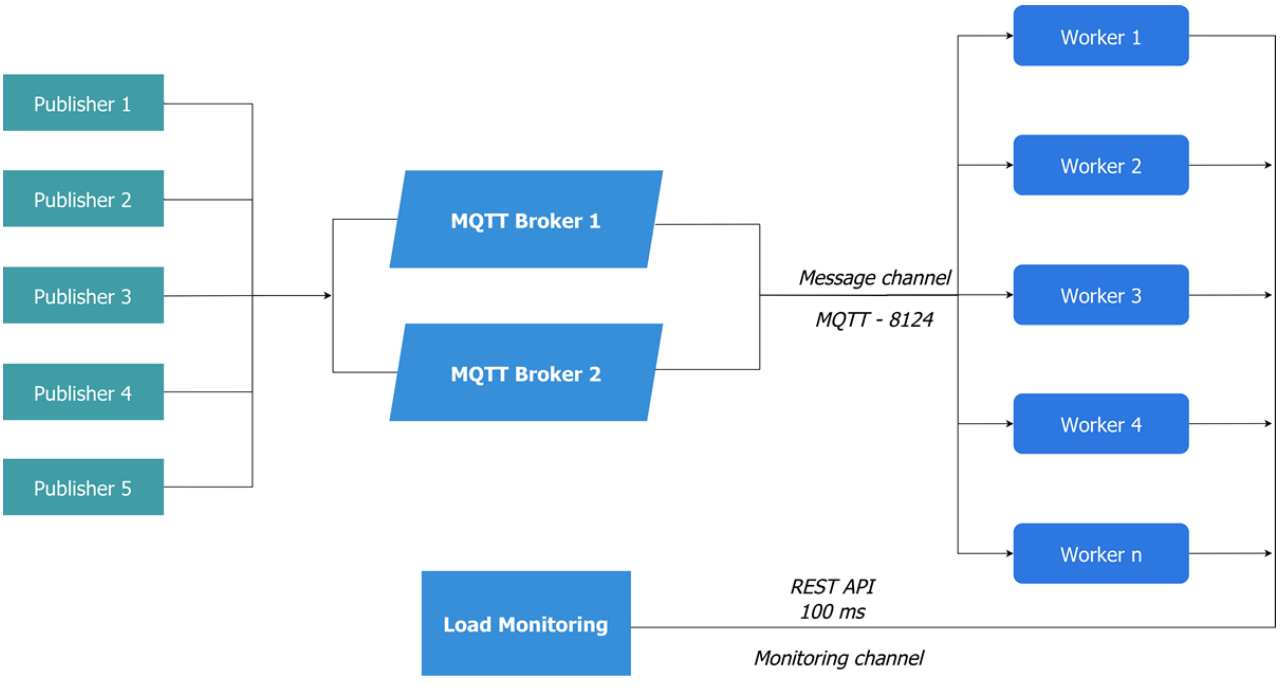


Fig. 6. Scheme of experiment

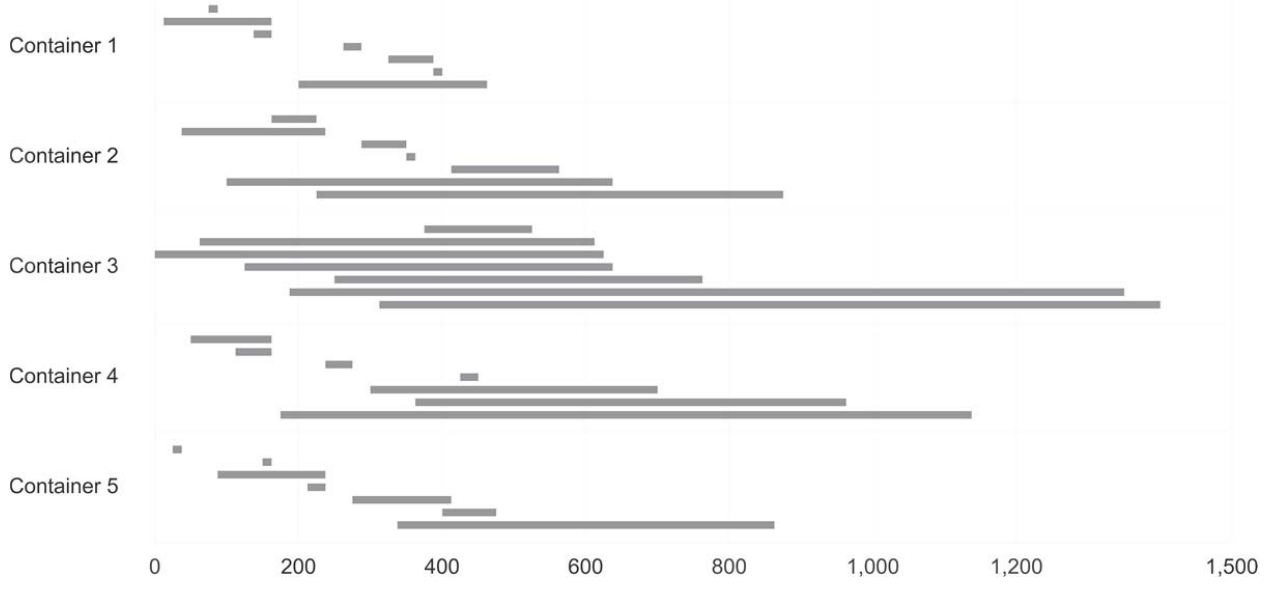


Fig. 7. Task execution time chart — Method 1

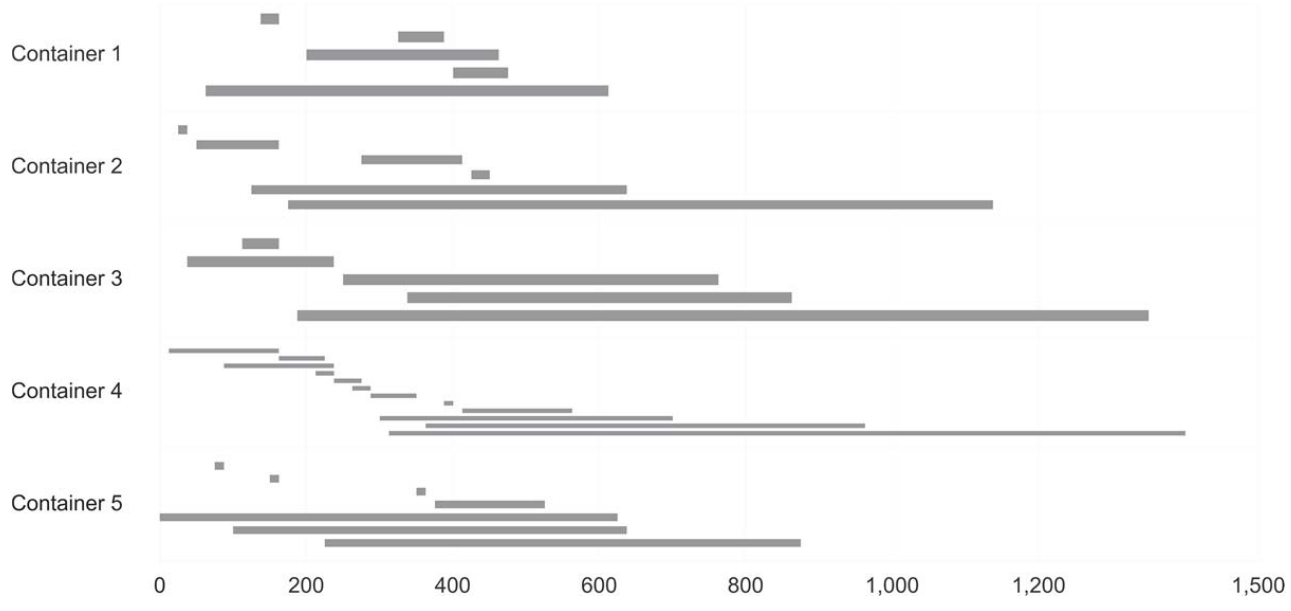


Fig. 8. Task execution time chart – Method 2

Table 4

Load balancing – Method 1

Task number	Server number	Execution time		
		Start	Finish	Execution duration
1	3	1691166950859	1691167000860	50001
2	1	1691166951847	1691166963848	12001
3	5	1691166952849	1691166953850	1001
4	2	1691166969850	1691166969850	16000
5	4	1691166954851	1691166963852	9001
6	3	1691166955853	1691166999853	44000
7	1	1691166956856	1691166957856	1000
8	5	1691166957857	1691166969857	12000
9	3	1691166958855	1691167001856	43001
10	1	1691166959856	1691166963857	4001

Table 5

Load balancing – Method 2

Task number	Server number	Execution time		
		Start	Finish	Execution duration
1	5	1691167138683	1691167188684	50001
2	4	1691167139682	1691167151683	12001
3	2	1691167140684	1691167141684	1001
4	3	1691167141687	1691167157688	16000
5	2	1691167142686	1691167151686	9001
6	1	1691167143691	1691167187692	44000
7	5	1691167144688	1691167145689	1000
8	4	1691167145695	1691167157695	12000
9	5	1691167146693	1691167189693	43001
10	3	1691167147694	1691167151695	4001

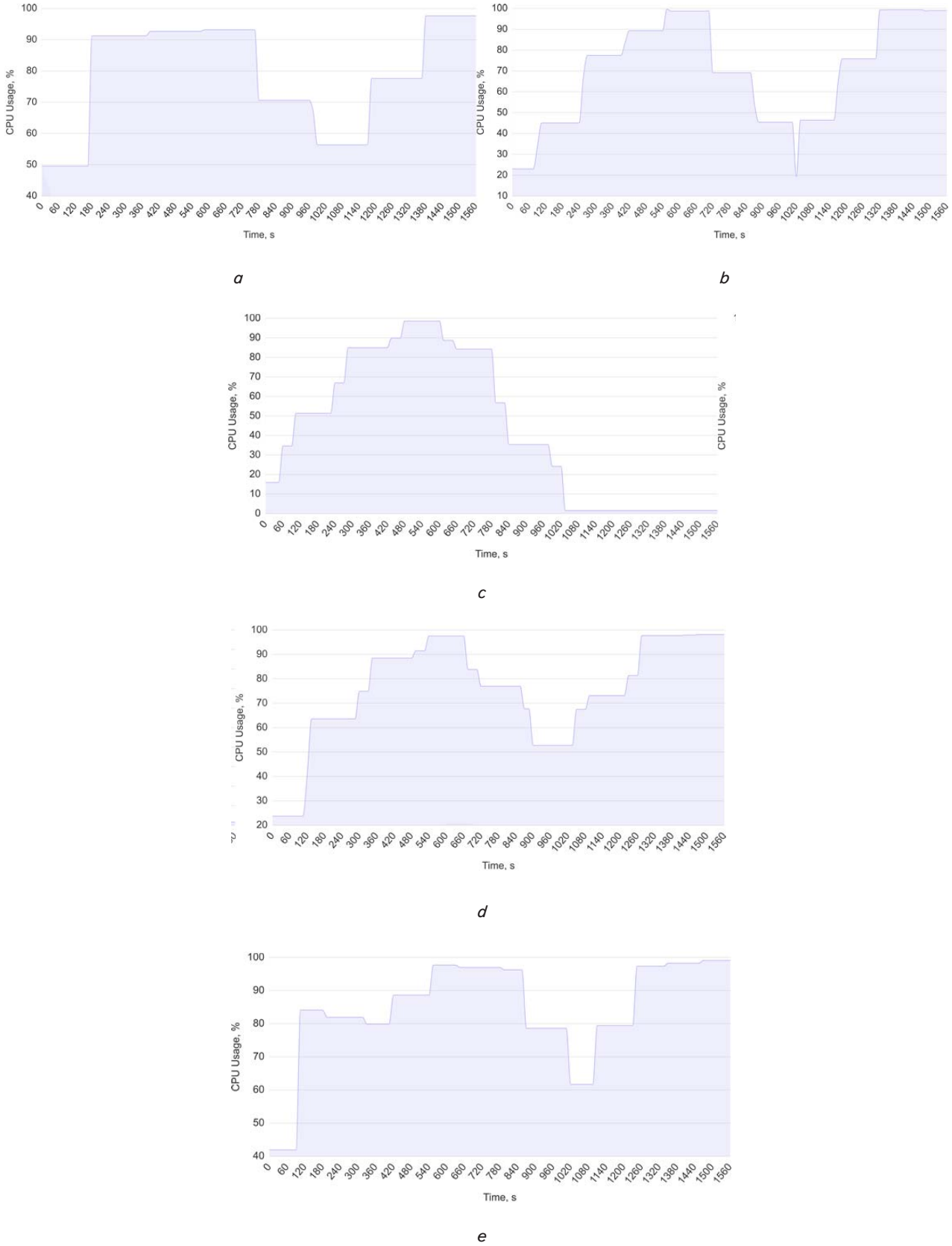
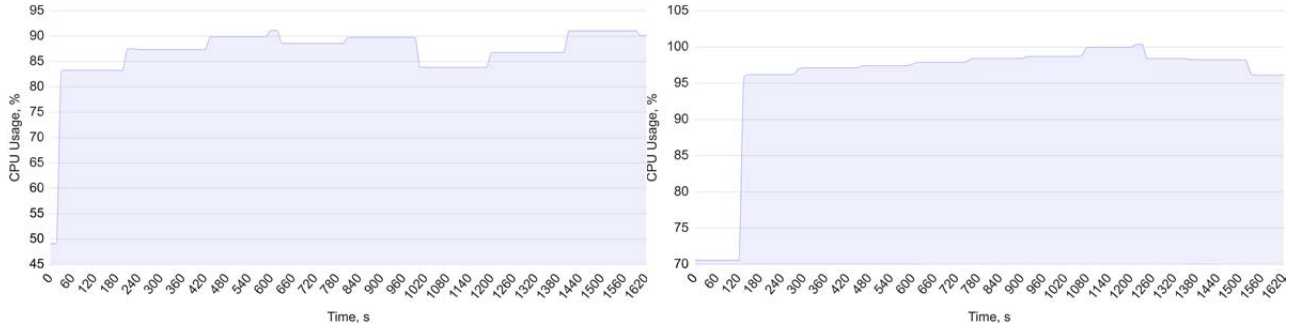
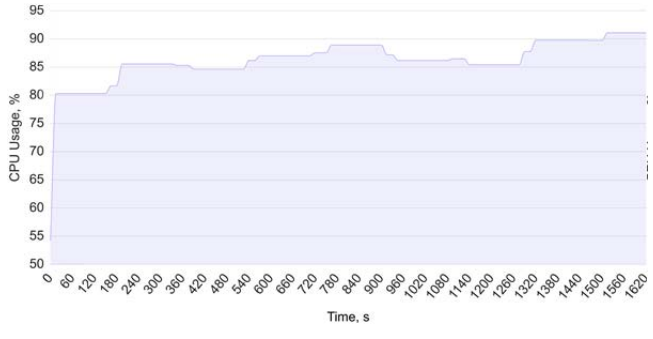


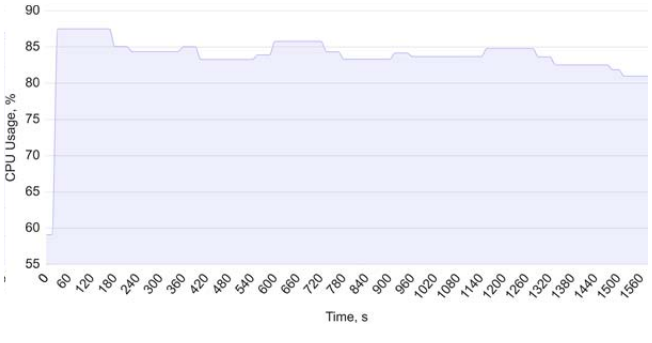
Fig. 9. Dynamics of changes in processor resource load. Method 1:
a — container; *b* — container 2;
c — container 3; *d* — container 4;
e — container 5



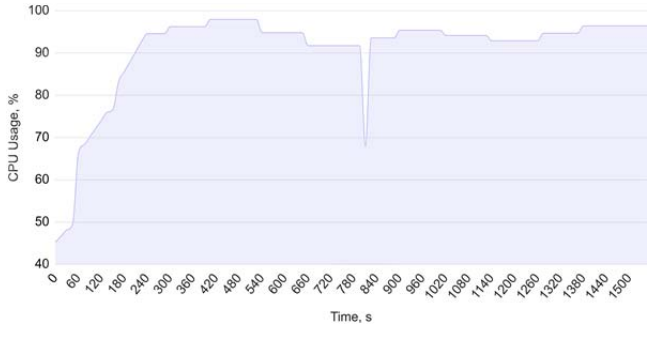
a *b*



c



d



e

Fig. 10. Dynamics of changes in processor resource load. Method 2:
a — container; *b* — container 2;
c — container 3; *d* — container 4;
e — container 5

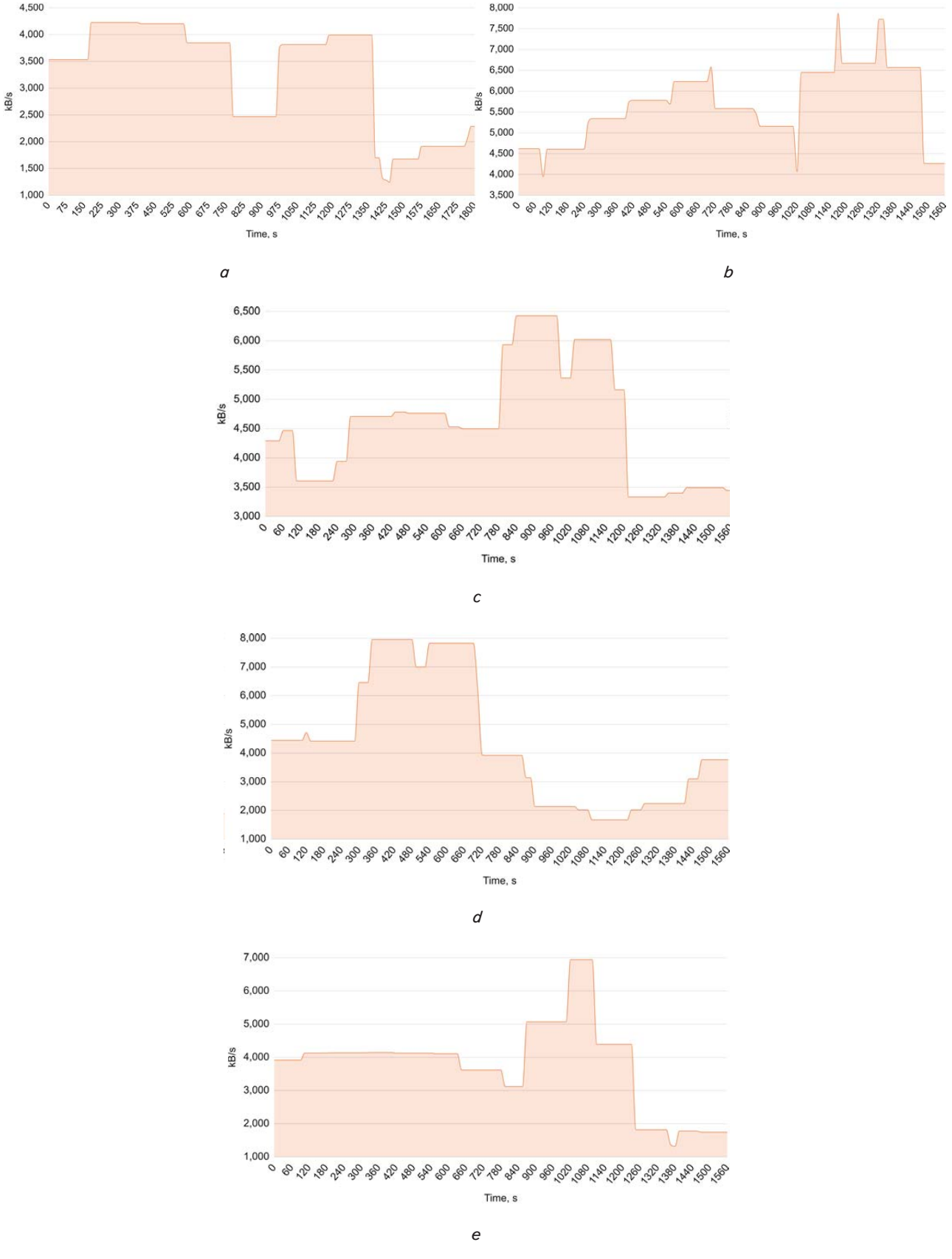
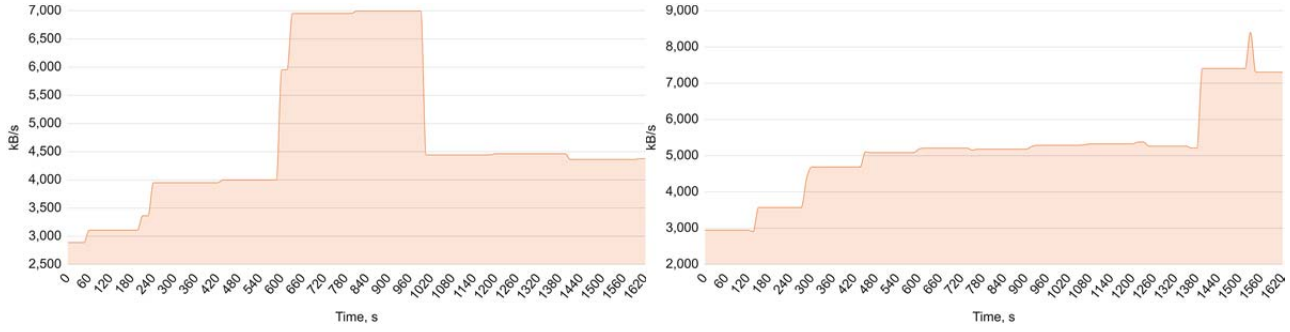
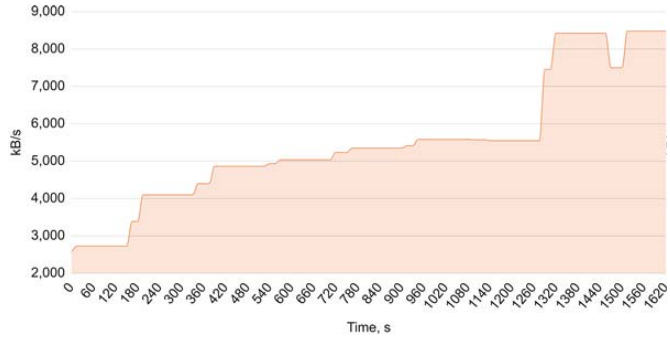


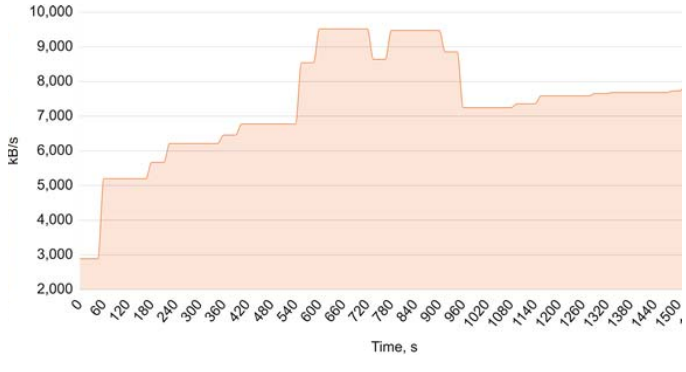
Fig. 11. Dynamics of changes in processor resource load. Method 1:
a — container; *b* — container 2;
c — container 3; *d* — container 4;
e — container 5



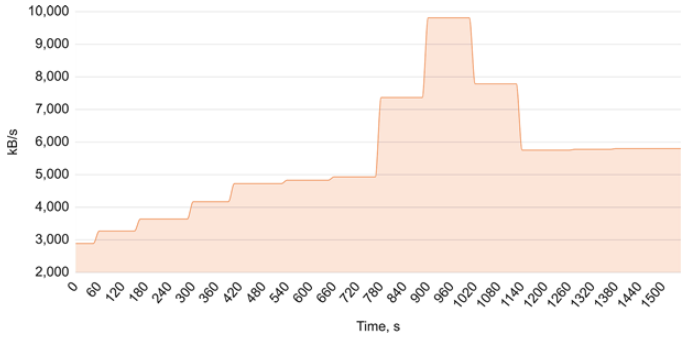
a *b*



c



d



e

Fig. 12. Dynamics of changes in processor resource load. Method 2:
a — container; *b* — container 2;
c — container 3; *d* — container 4;
e — container 5

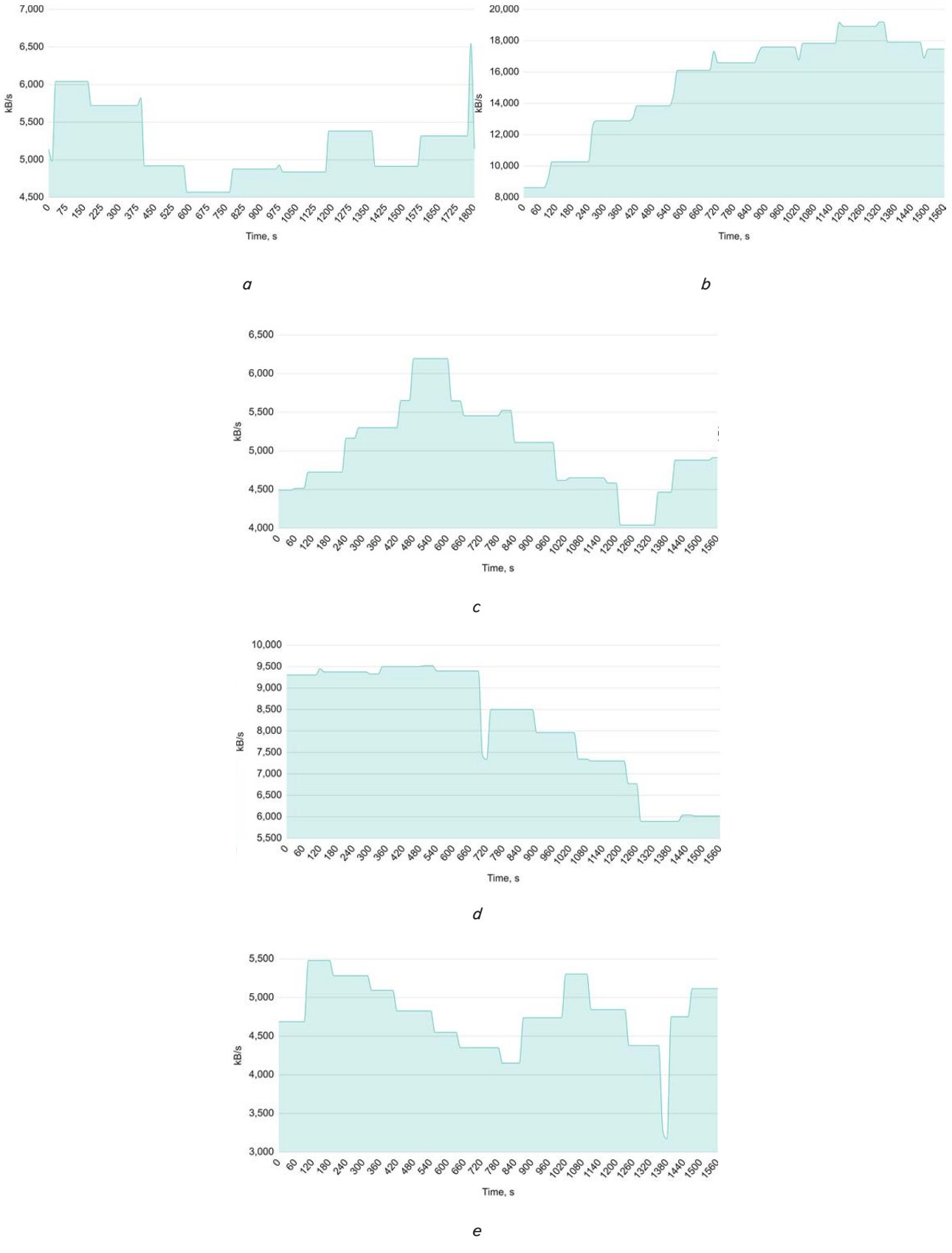


Fig. 13. Dynamics of changes in processor resource load. Method 1:
a — container; *b* — container 2;
c — container 3; *d* — container 4;
e — container 5

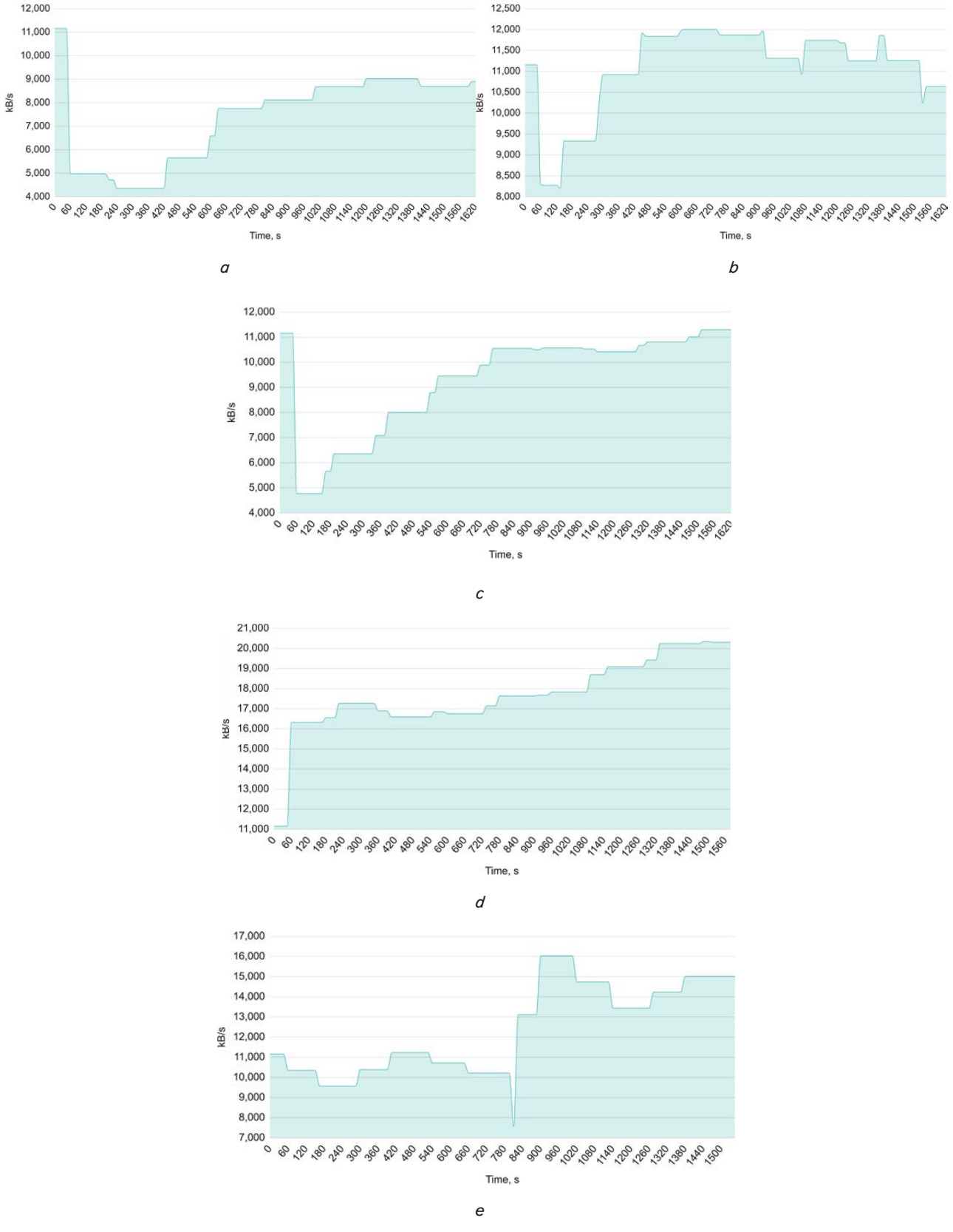


Fig. 14. Dynamics of changes in processor resource load. Method 2: *a* — container; *b* — container 2; *c* — container 3; *d* — container 4; *e* — container 5

For both studied methods, the load was generated during the same period of time and with the same intensity. Therefore, based on our results, it is possible to conduct a comprehensive comparison of their effectiveness.

5. 5. 3. Load distribution coefficient

The load distribution uniformity coefficient was determined according to the method described in chapter 5.4 for instantaneous server load values with an interval of 100 ms. An example of calculating the coefficient for 5 servers with the following load values:

Server 1: x_1={0.8, 0.7, 0.6, 0.5},

Server 2: x_2={0.9, 0.65, 0.55, 0.45},

Server 3: x_3={0.75, 0.6, 0.7, 0.55},

Server 4: x_4={0.85, 0.75, 0.65, 0.6},

Server 5: x_5={0.88, 0.68, 0.58, 0.48}. (5)

Euclidean distance between servers 1 and 2:

$$\text{Distance}(x_1,x_2)= =\sqrt{(0.8-0.9)^2+(0.7-0.65)^2+(0.6-0.55)^2+(0.5-0.45)^2}\approx \approx 0.169. \quad (6)$$

Similarly, the distances between all pairs of servers are calculated. The average distance can then be calculated as an aggregated metric:

$$\text{Average Distance}= =\frac{1}{5\times 4}\left(\begin{array}{l}\text{Distance}(x_1,x_2)+\\ +\text{Distance}(x_1,x_3)+\ldots+\\ +\text{Distance}(x_4,x_5)\end{array}\right)\approx 0.173. \quad (7)$$

The plot in Fig. 15 shows the values of the calculated load distribution coefficient in the system for two balancing methods. In this case, the smaller the value of the calculated distance, the more uniform the distribution, and accordingly, this method is more effective.

The above coefficient is the main indicator for analyzing and evaluating the performance of load balancing methods as it reflects the system's ability to adapt and optimize the distribution of tasks and resources. This indicator makes it possible to assess how well the system coped with the variable load, and how efficiently the available computing resources are used.

6. Discussion of research results of research on an improved load balancing method in distributed Internet of Things systems

The results of load distribution in the system reported in this work demonstrate the advantages of the proposed balancing procedure compared to the existing one – MQTT Shared Subscriptions [2]. This becomes possible owing to the proposed method of load balancing based on multi-parameter monitoring.

To implement the proposed methods and algorithms, an improved MQTT broker architecture (Fig. 4) with additional components was developed. Namely, the monitoring and load balancing module. In addition, the client's MQTT module has been expanded to determine and transmit the values of the current state of the investigated parameters (CPU Utilization, RAM Usage, Disk Usage, Network Utilization).

A mathematical model (1) and the corresponding coefficient are proposed to estimate the workload of the computing server. In this model, dynamic coefficients are used for the parameters on which the server is monitored. This approach makes it possible to adapt balancing according to the type of input load, which helps increase the evenness of the use of computing resources. This model is programmatically implemented in the broker's MQTT monitoring module. The module receives load vectors from active servers and calculates the value of the coefficient.

An algorithm (Fig. 5) for dynamic load balancing was developed, and its software implementation was performed in the corresponding MQTT broker module. This algorithm performs balancing based on load factors and threshold values for each server. In the software implementation of the algorithm, the value of server thresholds is static, and the value of load factors is dynamic and updated every 100 milliseconds.

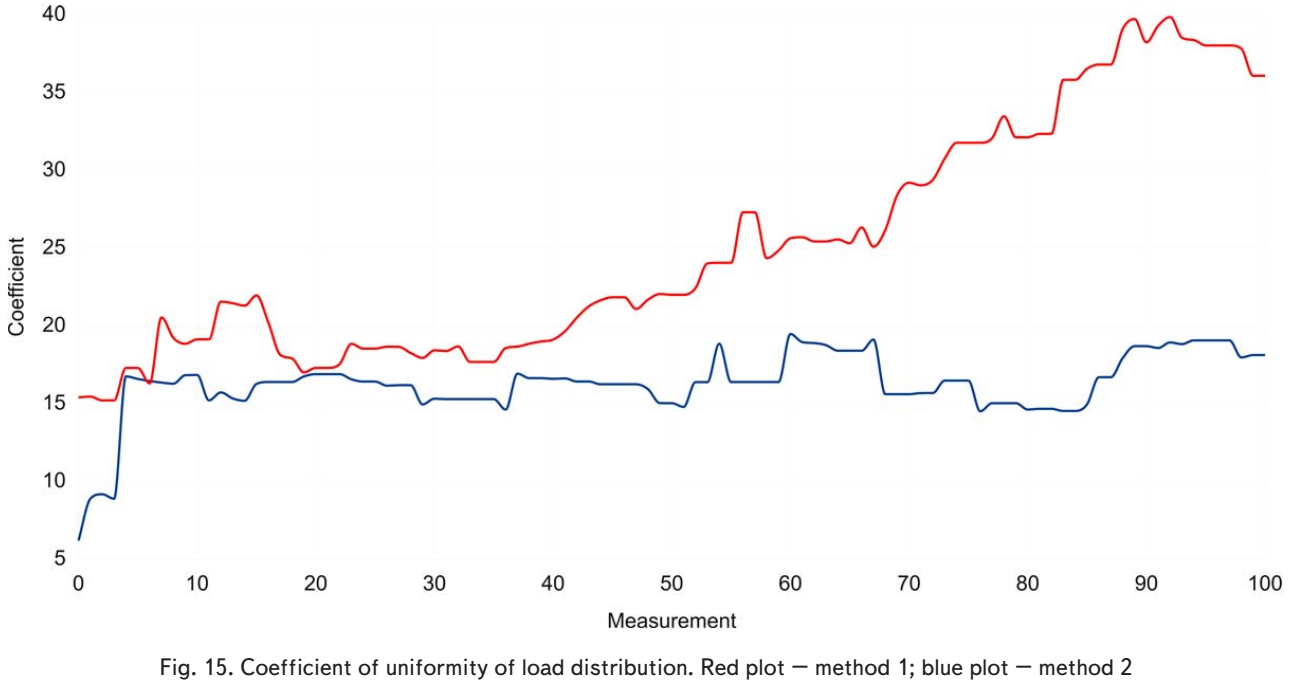


Fig. 15. Coefficient of uniformity of load distribution. Red plot – method 1; blue plot – method 2

To assess the uniformity of load distribution, a mathematical model (4) was built, based on the distances between the vectors of instantaneous load values of active servers. The advantage of this model is that it takes into account the differences in server load on several parameters (this study uses a model with four parameters). This approach allows for a comprehensive assessment of the efficiency of the use of computing resources and is simple to implement.

Our experiment with the simulation of a real IoT system that generates a dynamic load showed the main differences in the operation of these two methods.

From the analysis of the time chart in Fig. 7, one can see that using the standard Round Robin balancing algorithm (used in Method 1) leads to overloading of some servers and inefficient use of others. The situation when 7 tasks were executed simultaneously on the server Worker 3, while only one was executed on the server Worker 1, is especially revealing. This unevenness in the load occurs because the above algorithm does not take into account the current state of server load and the resources needed to process specific tasks, which may differ in complexity or volume of data.

Comparing this with Fig. 8, where the proposed algorithm is shown, one can see that it provides a more even distribution of tasks among active servers. The maximum difference in the number of active tasks during the entire testing period does not exceed one. This indicates that the proposed method demonstrates a more stable and adaptive approach to load balancing, ensuring optimal use of computing resources.

Analysis of the plots in Fig. 9–14 confirms the influence of the proposed methods in ensuring the effective use of the main resources of the system and demonstrates a significant advantage of the proposed method. This conclusion is confirmed by the plot in Fig. 15, which shows the change in the distribution uniformity coefficient for the considered methods. It is characteristic of the proposed method that the value of this coefficient is on average 70 % higher compared to Method 1, and does not change significantly during the experiment period. This ratio reflects the overall efficiency and projected performance of the system, and increasing it helps reduce resource costs as the system operates more efficiently and stably.

Our results confirm the proposed hypothesis as well as substantiate the application of the proposed methods to solve the problem of load balancing in distributed systems of the Internet of Things.

The limitation of the proposed approach is its focus on the MQTT protocol, and the impossibility of application in systems built on the basis of the HTTP, COAP, gRPC protocols.

The disadvantages of the proposed approach include the complicated MQTT architecture of the broker, as well as the need to provide an additional channel and a module for monitoring connected servers. A possible optimization option is to combine data transmission channels and a monitoring channel. Also, in this implementation, serialized JSON packets are used to transmit monitoring data. A more optimal solution may be the use of binary data transfer formats, for example, Protocol Buffers.

This research may be further advanced by adapting the proposed methods to new architectures and protocols that are used in distributed systems of the Internet of Things. Also, the development of this research may tackle the software optimization of the shortcomings indicated in this chapter, namely the improvement of the monitoring channel.

The methods and approaches proposed in this work could be applied to the development and improvement of distributed IoT systems and would make it possible to increase the productivity and efficiency of the use of computing resources.

7. Conclusions

1. An improved MQTT architecture of the broker, as well as connected clients (subscribers), has been implemented. In the implemented architecture, unlike the existing ones, additional components are introduced – the monitoring and load balancing module.

2. A mathematical model is proposed, which makes it possible to quantitatively estimate the instantaneous load of the server. In this model, unlike the existing models, the method of multi-parameter monitoring of the state of computing resources is applied. In addition, dynamic coefficients are applied for each parameter, which makes this model adaptable to different types of load. The model is an element of the proposed method, and is used in the above MQTT broker load balancing module.

3. An algorithm for dynamic load balancing based on the multiparameter monitoring method was developed. This algorithm provides an adaptive distribution of tasks among active computing servers and significantly reduces the risk of overloading and underutilization of computing resources compared to existing methods. The application of this algorithm has made it possible to improve the overall efficiency of the use of computing resources of the system by up to 70 %.

4. For a general assessment of the efficiency of load distribution in the system, a mathematical model based on Euclidean distances between vectors of instantaneous server load values is proposed. On the basis of this model, a special indicator was introduced – the coefficient of uniformity of load distribution. This coefficient, in contrast to existing assessment methods, reflects the balanced distribution of the load of the system as a whole, and not its individual components.

5. To determine the effectiveness of the proposed methods, a testing algorithm was developed, and a number of scientific and practical experiments were conducted. A comparative analysis of the effectiveness of balancing in a distributed system of the Internet of Things was carried out based on existing and proposed methods. The results of the experiment showed that the use of the proposed methods makes it possible to reduce the average load of the computer server by 40–65 %. At the same time, the speed of data processing remains unchanged. This means that the application of the suggested procedure does not affect the performance of the system but helps reduce the load on computing resources.

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 and the results reported in this paper.

Funding

The study was conducted without financial support.

Data availability

All data are available in the main text of the manuscript.

References

1. State of IoT – Spring 2023. Available at: https://iot-analytics.com/product/state-of-iot-spring-2023
2. Liaqat, M., Naveed, A., Ali, R. L., Shuja, J., Ko, K.-M. (2019). Characterizing Dynamic Load Balancing in Cloud Environments Using Virtual Machine Deployment Models. IEEE Access, 7, 145767–145776. doi: https://doi.org/10.1109/access.2019.2945499
3. Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. IEEE Access, 9, 41731–41744. doi: https://doi.org/10.1109/access.2021.3065308
4. Goncalves, D., Puliafito, C., Mingozzi, E., Rana, O., Bittencourt, L., Madeira, E. (2020). Dynamic Network Slicing in Fog Computing for Mobile Users in MobFogSim. 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC). doi: https://doi.org/10.1109/ucc48980.2020.00042
5. Yuan, H., Bi, J., Zhou, M. (2022). Geography-Aware Task Scheduling for Profit Maximization in Distributed Green Data Centers. IEEE Transactions on Cloud Computing, 10 (3), 1864–1874. doi: https://doi.org/10.1109/tcc.2020.3001051
6. Bogdanov, K. L., Reda, W., Maguire, G. Q., Kostić, D., Canini, M. (2018). Fast and Accurate Load Balancing for Geo-Distributed Storage Systems. Proceedings of the ACM Symposium on Cloud Computing. doi: https://doi.org/10.1145/3267809.3267820
7. Srinivas, J., Qyser, A. A. M., Reddy, B. E. (2015). Exploiting Geo Distributed datacenters of a cloud for load balancing. 2015 IEEE International Advance Computing Conference (IACC). doi: https://doi.org/10.1109/iadcc.2015.7154780
8. Shuaib, M., Bhatia, S., Alam, S., Masih, R. K., Alqahtani, N., Basheer, S., Alam, M. S. (2023). An Optimized, Dynamic, and Efficient Load-Balancing Framework for Resource Management in the Internet of Things (IoT) Environment. Electronics, 12 (5), 1104. doi: https://doi.org/10.3390/electronics12051104
9. Lim, J. (2021). Scalable Fog Computing Orchestration for Reliable Cloud Task Scheduling. Applied Sciences, 11 (22), 10996. doi: https://doi.org/10.3390/app112210996
10. Singh, S. P., Kumar, R., Sharma, A., Nayyar, A. (2020). Leveraging energy-efficient load balancing algorithms in fog computing. Concurrency and Computation: Practice and Experience, 34 (13). doi: https://doi.org/10.1002/cpe.5913
11. Fan, Q., Ansari, N. (2020). Towards Workload Balancing in Fog Computing Empowered IoT. IEEE Transactions on Network Science and Engineering, 7 (1), 253–262. doi: https://doi.org/10.1109/tnse.2018.2852762
12. Kim, H.-Y., Kim, J.-M. (2016). A load balancing scheme based on deep-learning in IoT. Cluster Computing, 20 (1), 873–878. doi: https://doi.org/10.1007/s10586-016-0667-5
13. Gomez, C., Shami, A., Wang, X. (2018). Machine Learning Aided Scheme for Load Balancing in Dense IoT Networks. Sensors, 18 (11), 3779. doi: https://doi.org/10.3390/s18113779
14. Adil, M. (2021). Congestion free opportunistic multipath routing load balancing scheme for Internet of Things (IoT). Computer Networks, 184, 107707. doi: https://doi.org/10.1016/j.comnet.2020.107707
15. Tonguz, O. K Yanmaz, E. (2008). The Mathematical Theory of Dynamic Load Balancing in Cellular Networks. IEEE Transactions on Mobile Computing, 7 (12), 1504–1518. doi: https://doi.org/10.1109/tmc.2008.66
16. Latchoumi, T. P., Parthiban, L. (2021). Quasi Oppositional Dragonfly Algorithm for Load Balancing in Cloud Computing Environment. Wireless Personal Communications, 122 (3), 2639–2656. doi: https://doi.org/10.1007/s11277-021-09022-w
17. Zakutynskyi, I. (2023). Finding the Optimal Number of Computing Containers in IoT Systems: Application of Mathematical Modeling Methods. Electronics and Control Systems, 2 (76), 9–14. doi: https://doi.org/10.18372/1990-5548.76.17661
18. Alakbarov, R. (2022). An Optimization Model for Task Scheduling in Mobile Cloud Computing. International Journal of Cloud Applications and Computing, 12 (1), 1–17. doi: https://doi.org/10.4018/ijcac.297102
19. Kaveri, P. R., Chavan, V. (2013). Mathematical model for higher utilization of database resources in cloud computing. 2013 Nirma University International Conference on Engineering (NUiCONE). doi: https://doi.org/10.1109/nuicone.2013.6780095
20. Zakutynskyi, I., Sibruk, L., Rabodzei, I. (2023). Performance evaluation of the cloud computing application for IoT-based public transport systems. Eastern-European Journal of Enterprise Technologies, 4 (9 (124)), 6–13. doi: https://doi.org/10.15587/1729-4061.2023.285514
21. MQTT Shared Subscriptions – MQTT 5 Essentials Part 7. Available at: https://www.hivemq.com/blog/mqtt5-essentials-part7-shared-subscriptions/
22. MQTT Version 5.0. OASIS Standard. Available at: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html