Structural Synthesis of the IoT System for the Fog Computing

Evgeny Saksonov, Yury Leokhin
Moscow Technical University for Communication and Informatics
Moscow, Russia
saksmiem@mail.ru, y.l.leokhin@mtuci.ru

Peter Panfilov
National Research University – Higher School of Economics
Moscow, Russia
ppanfilov@hse.ru

Abstract—**The paper presents the results of the structural design of the Internet of Things system, allowing to distinguish subsystems for cloud and fog computing to calculate parameters of information flows between them, and to support the structure optimization tasks.**

I. Introduction

The Internet of Things (IoT) is finding more and more application in various areas of human activity. According to experts, by 2020 the number of "things" connected to the Internet is estimated at tens of billions [1], [2], [3], [4]. All this dramatically increases the number of application tasks that can be solved using the IoT technology and, accordingly, the volume of traffic transmitted in the IoT systems.

As a rule, the structure of the IoT system includes several levels: at a lower level, the controllers receive data from sources ("things"), and at the higher levels, the data processing occurs (storage, application execution, management tasks, etc.) [2]. Taking into account the increase in the number of controllers in a system, the traffic over communication channels between the lower and higher levels in a system over the time will become an obstacle to the development of such a systems. It is advisable to transfer part of the data processing tasks to the lower level. This is possible by increasing the total computing power of IoT controllers when combining these in a single distributed computing environment.

In 2014, the IoT World Forum (IWF) published a reference model of the Internet of Things [5], where it is recommended to use the computing power of controllers directly connected to data sources in data processing (application execution). For this purpose, it is necessary to create a computing environment consisting of controllers and communication links between them. Such an approach is called "fog computing".

A somewhat more radical solution is to establish a level of peripheral (edge, boundary) data processing also known as "edge computing", when the incoming data from lower level sources is processed on site, and the results are forwarded to the higher level(s). In such an arrangement, a cloud computing is used by complex applications that require information from many spatially distributed IoT devices.

This approach conforms to the selection of general-purpose devices in the ITU-T model (ITU-T) [6], [7].

This way, it makes it possible to reduce the data traffic transmitted between devices in the IoT system at lower level and applications in the cloud at higher level, and to reduce the amount of data stored at the higher level. This allows to significantly increase the computing performance (especially if the cloud infrastructure is at a distance from the IoT devices), and to provide application problem solving in real time.

II. Problem statement

There are some restrictions on practical application of fog and edge computing that are associated with the limitations of the IoT controllers in terms of performance, memory size and other parameters. In this regard, a demand for computing at higher levels will remain and even increase with an increase in size of the IoT systems necessary for solving the problem, in required computing power, in data storage size and other parameters. Also the problem of migrating some applications and traffic between them to the lower levels will also remain and will be increasingly relevant.

It is required to design the structure of the system in such a way so to coordinate the capacities of the controllers and communication channels with the requirements for the characteristics of the tasks to be performed. Thus, there exists a problem of designing the structure of the IoT system, that is maximally adapted to a special class of tasks and that of a quantitative assessment of the proposed designs.

The structural design implies the selection of a set of the interacting processes (applications) of a solution of a task and its distribution between controllers and a cloud so that to lower down the data traffic in communication channels, and to transfer computational load to controllers.

Solutions of similar problems can be found, for example, in our previous works [13], [14], where the design and analysis of computer networks is considered. In this research, we seek for the formal solution to the problem of structural design of the IoT system that makes it possible to configure and analyze complex distributed systems with fog, edge and cloud computing components and that minimizes data traffic to the cloud during the operation of the IoT system.

II. Related Works

The graph partition problem is well known in mathematics [8]. Given a graph **G**={**X**, **R**}, where **X** denotes the set of *n* vertices and **R** the set of edges, a (*k*, *v*) balanced partition problem, is the task of partitioning **G** into *k* components of at most size *v* (*n*/*k*), while minimizing the capacity of the edges between separate components [9]. Also, given graph **G** and an integer $k > 1$, a bicriteria-approximation approach considers a task of partitioning **X** into *k* disjoint and equal sized parts (subsets), so that the number of edges between different parts is minimized. The problem further can be extended to the so called hypergraphs where edge can connect more than two vertices. When all vertices are in one partition, a hyperedge is not cut, and cut exactly once otherwise, no matter how many vertices are on each side.

Graph partition problems represent a class of so called NP-hard problems that require use of heuristics and approximation algorithms to derive applicable solutions. A good survey on recent trends in computational methods for solving balanced graph partitioning problems and applications can be find in [10]. At most, solutions presented in literature offer algorithms for graph partitioning using specific graph metrics and criteria (e.g., while minimizing total weight of edges between separate component subgraphs), and do not allow for easy quantitative evaluation of results for given variant of partition [11], [12].

In this paper, we describe an approach that would allow for quantitatively evaluating quality of the graph partition solution using different criteria based on weights of edges and/or vertices in a source graph. This approach would be applicable to evaluate solution quality while applying a graph partitioning algorithm as well as evaluating volumes of data flows between subsets of vertices in a source graph [8].

III. Mathematical model. General results

Suppose we have a problem defined on data represented in the form of a directed graph **G**={**X**, **R**}. Here $\mathbf{X} = \{x_i\}$ (i = 1, 2, …, N) is the set of vertices, $\mathbf{R} = \{r_{ij}\}$ (i , j = 1, 2, …, N) is the set of edges. Both sets are not empty and contain a finite number of elements ($0 < N \leq \infty$). The graph is coherent [8].

Each vertex and each edge of the graph has its own weight. The vertex v_i's weight is x_i, where $\infty > v_i \geq 0$; edge r_{ij}'s weight is λ_{ij} , where $\infty > \lambda_{ij} \geq 0$, for i, j = 1, 2, ..., N. The set of weights of the graph vertices is given by a vector $\mathbf{v} = (v_1, v_2, ..., v_N)$. The set of weights of the edges of a graph is given by the matrix $\mathbf{\Lambda}_0 = \|\lambda_{ij}\|$, ($i, j$ = 1, 2, ..., N). The matrix has zero (nonzero) elements that coincide with zero (nonzero) elements of the adjacency matrix of the graph **G**.

The set of vertices **X** of a graph **G** is divided into disjoint non-empty subsets $\mathbf{x}_1$, $\mathbf{x}_2$, …, $\mathbf{x}_K$ so that $\bigcup_{j=1}^{K} \mathbf{x}_j = \mathbf{X}$, where K ($1 \leq K \leq N$) is the number of subsets in the partition of the set **X**.

The partition is specified by the matrix **L** of partitioning the set **X** into subsets of vertices: $\mathbf{L}(\mathbf{X}) = \|l_{mn}(\mathbf{X})\|$, ($m$ = 1, 2, ..., N; n = 1, 2, ..., K).

For all elements of the matrix **L (X)**, the following conditions hold:

- $l_{mn}(\mathbf{X}) = 1$, if the vertex m is in the n^{th} subset ($x_m \in \mathbf{X}_n$), and $l_{mn}(\mathbf{X}) = 0$, if the vertex m is not in the n^{th} subset of **X** ($x_m \notin \mathbf{X}_n$);
- $1 < \sum_{m=1}^{N} l_{mn}(\mathbf{X}) \leq N$, for any n = 1, 2, ..., K, i.e. each subset necessarily includes at least one vertex;
- $\sum_{n=1}^{K_0} l_{mn}(\mathbf{X}) = 1$, for any m = 1.2, …, N, i.e. each top can be a part only of one subset;
- $\sum_{n=1}^{K_0} l_{mn}(\mathbf{X}) = 1$, an equality is true for any partition, i.e. all vertices in **G** are distributed among subsets.

The column j of the matrix **L(X)** specifies the composition of the subset $\mathbf{x}_j$, (j = 1, 2, …, K).

The listed conditions allow to uniquely define the partition by specifying the matrix **L(X)**.

After splitting the set of vertices **X**, each obtained subset is considered a vertex in a new oriented graph $\mathbf{G}_1(\mathbf{X}_1, \mathbf{R}_1)$, where the set of vertices $\mathbf{X}_1$ contains K elements (x_{11}, x_{12},…, x_{1K}) whose second indices are their numbers in graph $\mathbf{G}_1$ that match the numbers of the corresponding subsets of which vertices are formed. The set of edges $\mathbf{R}_1$ ($\mathbf{R}_1 = \{r_{1ij}\}$) is the set of edges between the vertices x_{1i} и x_{1j} of graph $\mathbf{G}_1$, the weights of which are formed as sums of the weights of the edges between the vertices of graph **G**, which are part of the subsets $\mathbf{x}_i$ and $\mathbf{x}_j$, and have corresponding direction, for i , j = 1, 2, ..., K.

After partitioning, a new matrix of weights of the edges $\mathbf{\Lambda}_1$ of the graph $\mathbf{G}_1$ is calculated according to the formula:

$$\mathbf{\Lambda}_1 = \mathbf{L}^T \mathbf{\Lambda}_0 \mathbf{L} , \tag{1}$$

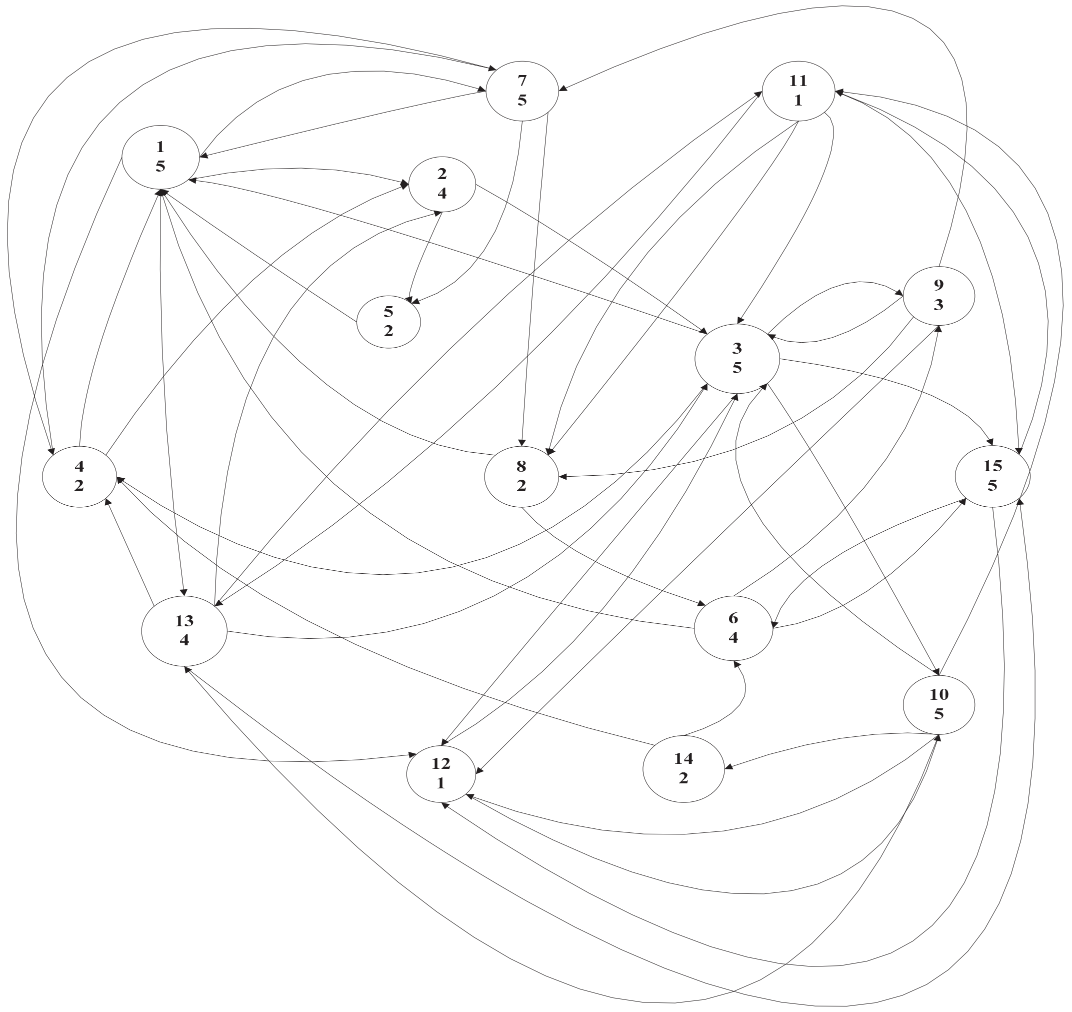
where $\lambda_{1ij} = \sum_{m=1}^{N} l_{mi} (\sum_{k=1}^{N} \lambda_{mk} l_{kj})$, ($i$, j = 1, 2, …, K).

The matrix $\mathbf{\Lambda}_1 = \|\lambda_{1ij}\|$, ($i, j$ = 1, 2, …, K) has zero (nonzero) elements that coincide with zero (nonzero) elements of the adjacency matrix of graph $\mathbf{G_1}$.

It follows from the formula (1) that for $i \neq j$, λ_{1ij} is the edge's weight between the vertices x_{1i} and x_{1j} in graph $\mathbf{G_1}$ and λ_{1ii} is the total weight of the edges between the vertices in graph **G** entering the subset $\mathbf{x}_i$, for i , j = 1, 2, …, K.

The weight of vertex x_{1j} in graph $\mathbf{G_1}$ is:

$$v_{1j} = \sum_{k=1}^{N} v_k l_{kj} , (j = 1, 2, ..., K). \tag{2}$$



$$\mathbf{\Lambda}_0 = \begin{pmatrix}
0 & 2 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 1 & 3 & 0 & 0 \\
0 & 0 & 4 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 3 & 7 & 0 & 4 & 0 & 0 & 5 \\
2 & 1 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 5 \\
2 & 0 & 0 & 1 & 2 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 3 & 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 2 \\
0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 4 \\
0 & 0 & 0 & 2 & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 2 & 2 & 0 & 0 & 0
\end{pmatrix}
\mathbf{L} = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}, \quad
\mathbf{\Lambda}_1 = \begin{pmatrix}
0 & 0 & 3 & 9 & 1 & 4 & 0 & 0 \\
4 & 22 & 3 & 4 & 7 & 0 & 0 & 3 \\
0 & 12 & 0 & 2 & 0 & 0 & 0 & 0 \\
6 & 12 & 0 & 4 & 0 & 3 & 2 & 2 \\
0 & 9 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 4 & 0 & 0 & 2 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 2 & 3 & 0 & 0
\end{pmatrix}.$$

$$\mathbf{v}_1 = (7, 16, 4, 12, 1, 5, 2, 3)$$

Fig. 1. Graph **G = {X, R}**

The weights of the vertices in graph **G₁** are given by the vector $\mathbf{v}_1 = (v_{11}, v_{12}, ..., v_{1K})$.

Thus, the formulas (1) and (2) allow for calculating the edge weights in graph **G₁** as well as the total edge capacity between the vertices in graph **G** included in the "combined" vertices in graph **G₁**.

Example 1. Let the graph **G** is **G**={**X**,**R**}, where number of vertices is $N = 15$. The sample graph **G** is presented in Fig. 1, where each vertex is marked with a column of its attributes, with the first value representing the vertex number, and the second value for a vertex' weight. Also the matrix $\mathbf{\Lambda}_0 = \left\|\lambda_{ij}\right\|$, ($i, j$ = 1, 2, ..., 15) of weights of edges of a graph **G**, the matrix $\mathbf{L}(\mathbf{X}) = \left\|l_{mn}(\mathbf{X})\right\|$, ($m$ = 1, 2, ..., N; n = 1, 2, ..., K) of the graph **G** partitioning, the matrix $\mathbf{\Lambda}_1 = \left\|\lambda_{1ij}\right\|$, ($i, j$ = 1, 2, …, K) of weights of the edges of the graph **G**₁, and the weight vector **v** of the vertices in graph **G**₁ are shown.

The corresponding graph **G**₁(**X**₁, **R**₁) is shown in Fig. 2, where edge weight is indicated at the beginning of each edge. For each vertex in graph **G**₁, the vertices of graph **G** are shown which are included in it. Each vertex in graph **G**₁ is marked with a column of three values (shown inside a vertex). The first value shows the number of the vertex. The second value is the sum of weights of the edges of graph **G** included in the vertex. The third one is the weight of the vertex as the sum of the weights of the vertices in graph **G** that constitute it.

The resulting model allows for calculating quantitative characteristics of the graph partition with various criteria.

Among the partitioning tasks with different criteria, the following ones may be of a special practical value:

1) **Task 1.** Finding graph partition with the minimum total weight of edges connecting the vertices in graph **G₁**.

Given a graph **G (X, R)** and matrix **Λ**₀ of the graph, find

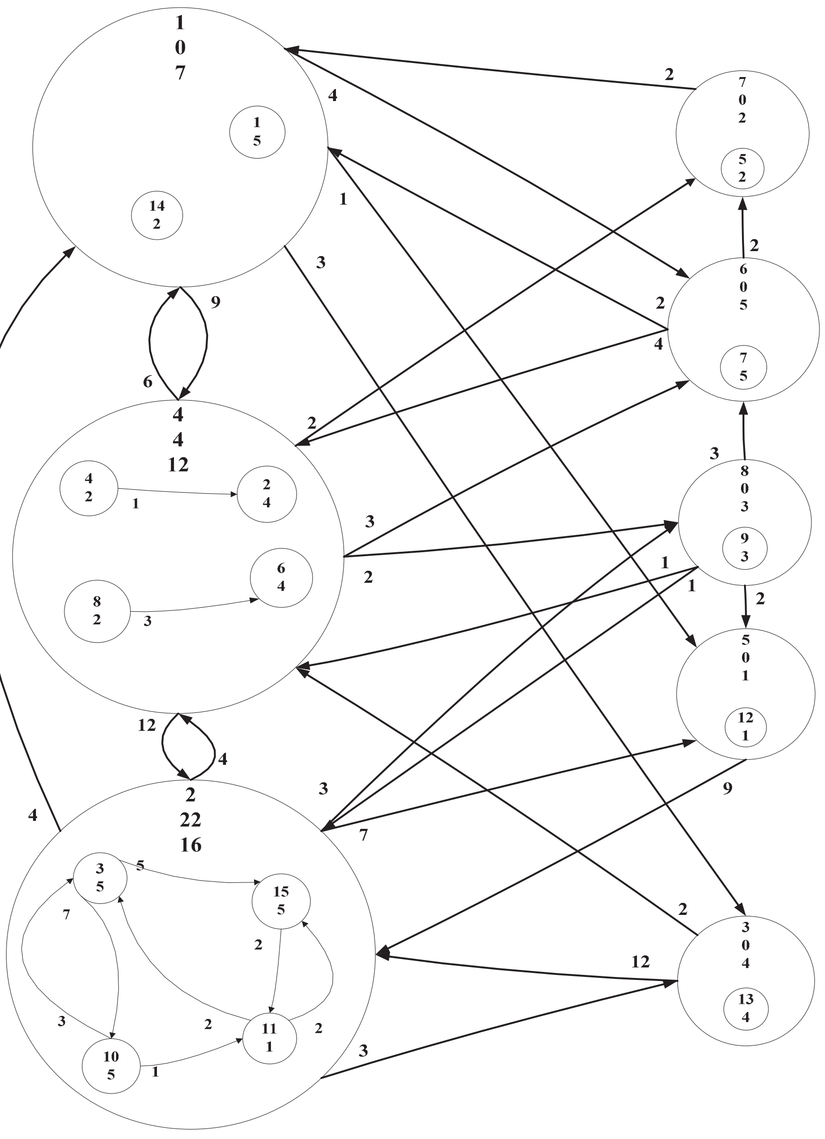


Fig. 2. Graph **G**₁(**X**₁, **R**₁)

$\min\limits_{K,\mathbf{L}}(\sum\limits_{i=1}^{K}\sum\limits_{j=1}^{K} r_{1ij}(\mathbf{L}))$.

2) **Task 2.** Finding graph partition with the maximum possible total weight of the vertices of graph $\mathbf{G_1}$.

Given a graph **G (X, R)**, a weight vector **v** of this graph, and the set $\overline{\mathbf{v}} = (\overline{v}_1, \overline{v}_2, ..., \overline{v}_K)$ of valid weights of the vertices in graph $\mathbf{G_1}$, find $\max\limits_{K,\mathbf{L}}(\sum\limits_{i=1}^{K} v_{1i}(\mathbf{L}))$, $v_{1k}(\mathbf{L}) \le \overline{v}_k$, for $k = 1,2, ..., K$.

As a rule, problems are solved under certain conditions on the parameters of graph $\mathbf{G_1}$. Such conditions may be as follows:

- restrictions on the maximum weight of the edges of graph $\mathbf{G_1}$: $\lambda_{1mn} \le \overline{\lambda}_{1mn}$, $(m, n = 1, 2, ..., K)$;
- restrictions on the maximum weight of the vertices of graph $\mathbf{G_1}$: $v_{1j} \le \overline{v}_{1j}$, $(j = 1, 2, ..., K)$;
- restrictions on the number of vertices in graph $\mathbf{G_1}$: $K \le \overline{K}$.

It is often advisable to solve the problems of obtaining not necessary optimal, but rather an "acceptable" solutions when partitioning a graph, that ensure the fulfillment of specific constraints on the quality of solution.

IV. Practical application of the modeling results

Practical application of the presented modeling results to the structural design of the IoT system makes it possible to configure systems with fog and edge computing components and conduct its analysis under conditions of traffic minimization to the cloud during the operation of the system. The solutions to similar problems in analyzing computer networks can be found in literature [13], [14].

In our approach, the graph **G = {X, R}** is the graph of the problem being solved. The vertices of the graph correspond to the applications, which are executed for solving specific tasks. It is assumed, that applications can run independently. The application number is the same as the vertex number in graph. Applications can exchange data (information). This data transfers are represented with the edges of the graph.

Directed graph defines the sequence of application execution tasks and directions of data transfer between applications.

The vertex x_i's weight v_i, for $i = 1, 2, …, N$, is a generalized parameter of the application (process), which shows what controller resources are needed for its execution (memory, processor performance, etc.), and edge r_{ij}'s weight λ_{ij} indicates the intensity of the data transmission (data traffic) between applications i and j ($\infty > \lambda_{ij} \ge 0$), $(i, j = 1, 2, ..., N)$.

For the fog and edge computing, communication between the controllers is usually organized via a separate local area network (LAN). With an increase of a number of interacting controllers, a special networking solutions are required, such as those provided by the new industrial IoT technologies.

When splitting the original task graph into subsets of processes (applications), the problem arises of distributing these subsets (applications) among real controllers at the lower level and the cloud servers at the higher level.

Example 2. Let's consider the graph **G = {X, R}** from example 1.

For this graph, a partition task 2 is solved with maximum total weight of vertices criteria. The number of vertices in the graph is 6, the maximum allowable weights of the vertices are as follows: $\overline{\mathbf{v}} = (55,7,7,5,7,5)$. The partitioning matrix **L** of the graph **G** and the weight matrix $\mathbf{\Lambda}_1$ take the following forms:

$$\mathbf{L} = \begin{pmatrix} 0&0&0&0&0&1\\ 0&1&0&0&0&0\\ 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 0&0&0&0&1&0\\ 0&0&1&0&0&0\\ 0&0&0&1&0&0\\ 0&0&1&0&0&0\\ 0&0&0&0&1&0\\ 1&0&0&0&0&0\\ 1&0&0&0&0&0\\ 1&0&0&0&0&0\\ 1&0&0&0&0&0\\ 0&0&0&0&1&0\\ 1&0&0&0&0&0 \end{pmatrix} \quad \mathbf{\Lambda}_1 = \begin{pmatrix} 53&3&3&0&4&3\\ 4&1&0&3&2&2\\ 8&0&3&0&2&4\\ 0&1&3&0&2&2\\ 3&2&6&3&0&2\\ 4&2&0&4&0&0 \end{pmatrix}$$

For this graph **G**, the solution to the task 2 is the graph $\mathbf{G}_2(\mathbf{X}_2, \mathbf{R}_2)$, as it is shown in Fig. 3.

The result obtained makes it possible to compose and analyze the structure of the IoT system, designed specifically for solving the distributed application problem using fog, edge and cloud computing. It allows for determining the characteristics of data transfers in communication channels (networks) and the workload on the controllers and the cloud.

Fig. 4 illustrates the variant of the network structure for the IoT system. It shows the controllers of the IoT system and the servers in the cloud, the total weight of the applications that run on the controllers and the cloud, the total intensities of data transfers (data traffic) over the network of controllers and the communication channel to the cloud. Options for building a network of controllers for organizing fog computing are given, as well.

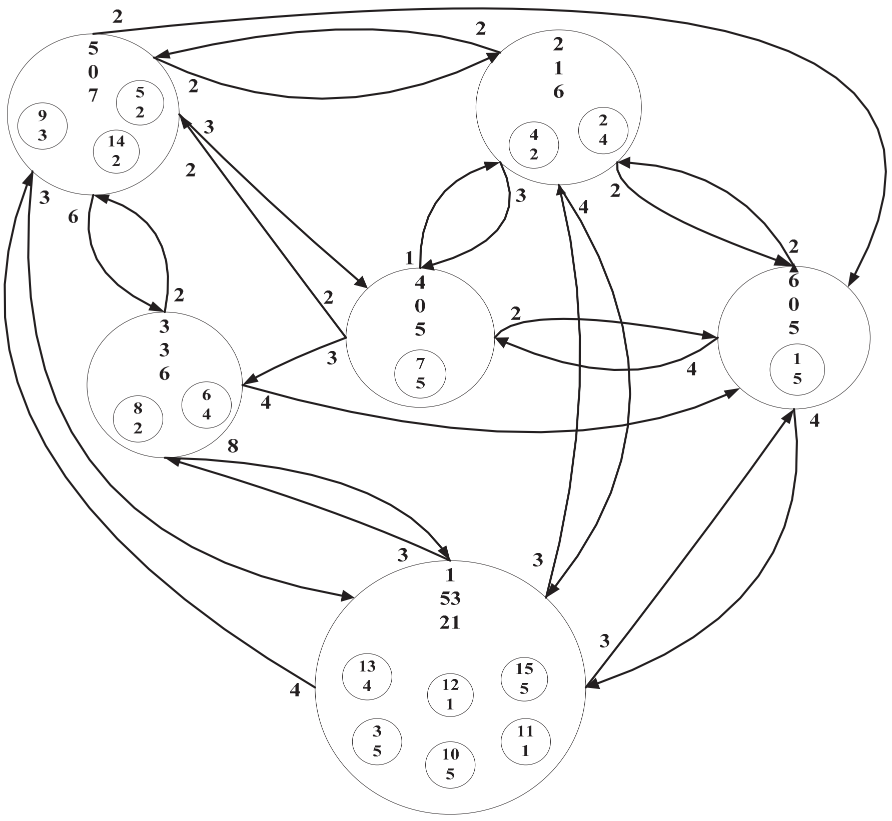


Fig. 3. Graph $\mathbf{G}_2(\mathbf{X}_2, \mathbf{R}_2)$

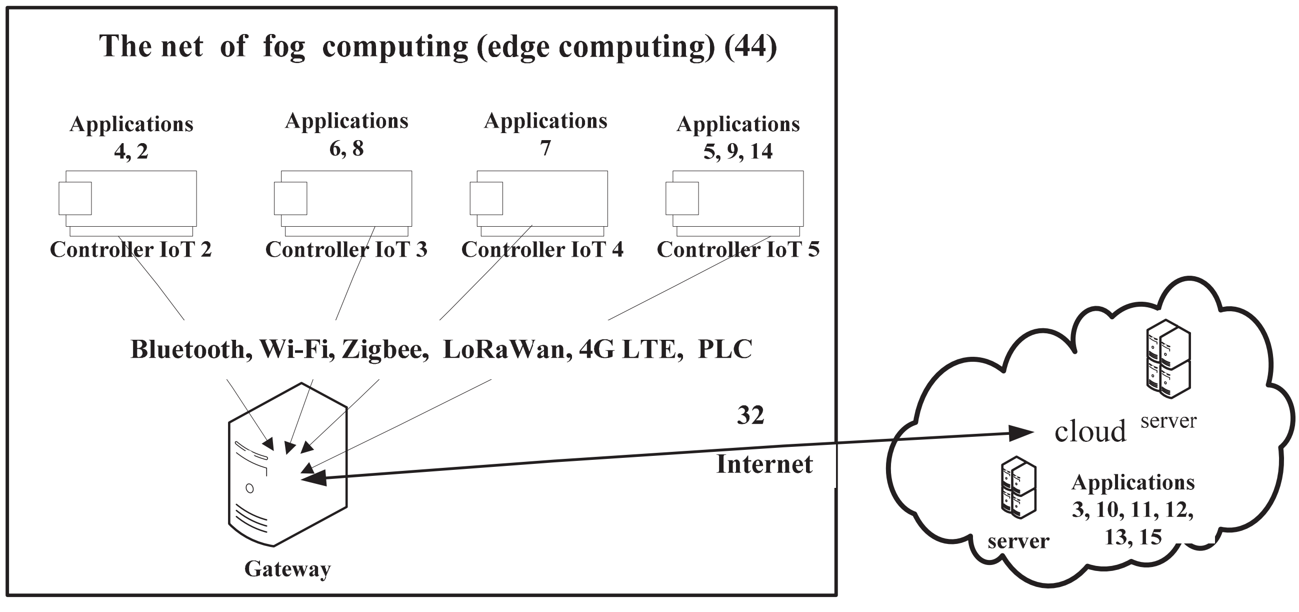


Fig. 4. Optional system structure

V. CONCLUSION

The development of the Internet of Things and related technologies and the increase in the number of interacting devices on the Internet of Things will require an increasing data processing at the level of fog and edge computing. The need to harmonize the performance of distributed applications and to integrate the results obtained at various levels will only increase, that will require a quantitative assessment of the made decisions while shaping the structure of the IoT systems.

We propose a formal approach to the structural design of an IoT system that allows for specifying the structure of a system with fog and edge computing components, and for solving design problems or structural optimization tasks under various conditions, as well as for determining the quantitative characteristics of the system under development.

In the core of the approach, a formal framework is provided for quantitatively evaluating quality of the graph partition solution using different criteria based on weights of edges and/or vertices in a source graph. This framework is applicable to evaluate solution quality while applying a graph partitioning algorithm as well as evaluating volumes of data flows between subsets of vertices in a source graph.

Further development of the proposed approach would include the development of the analytical techniques for multi-level graph splitting with a hierarchical structure.

REFERENCES

[1] J. Stankovic, “Research Directions for the Internet of Things”, *Internet of Things Journal,* Vol. 1, No. 1, 2014.

[2] W. Stallings, “The Internet of Things: Network and Security Architecture”, *The Internet Protocol Journal,* Vol. 18, No 4. 2017.

[3] A. McEwen, and H. Cassimally, *“Designing the Internet of Things”,* ISBN-13: 978-1118430620, Wiley, 2013.

[4] R. Khan, et al., “Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges”*, Frontiers of Information Technology (FIT),* 2012 10th International Conference on, 2012, pp. 257 - 260.

[5] Cisco Systems, “The Internet of Things Reference Model,” White Paper, 2014. Available at: *http://www.iotwf.com/.*

[6] ITU-T, “Overview of the Internet of Things,” Recommendation Y.2060, June 2012.

[7] ITU-T, “Common Requirements and Capabilities of a Gateway for Internet of Things Applications,” Recommendation Y.2067, June 2014.

[8] N. Cristofides, “Graph Theory an algorithmic approach”. *Management Science Imperial College*, London. Academic Press, New York, London, San Francisco. 1975.

[9] K. Andreev, H. Räcke.”Balanced Graph Partitioning”, *Proceedings of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures*. Barcelona, Spain. pp.120–124. ISBN 978-1-58113-840-5. doi:10.1145/1007912.1007931.

[10] A. Buluc, H. Meyerhenke, I. Safro, P. Sanders, C. Schulz. "Recent Advances in Graph Partitioning". arXiv:1311.3144.

[11] V.K. Pogrebnoy. “Matrix algorithm for graph partition problem”, *News of the Tomsk Polytechnic University*, Vol.310, 2007, pp.91-96. (in Russian)

[12] A.S. Shandrikov. “Sequential method of graph partitioning with external links minimizations”, *Scientific Herald of Vitebsk state university of technology*, Issue 5, 2003, pp.94-100. (In Russian)

[13] V.N. Azarov; E.A. Saksonov; Yu.L. Leokhin, “Analysis of Information Structure of the Corporate Network of Enterprise”, *Quality Management, Transport and Information Security, Information Technologies,* 2018 IEEE International Conference on, (IT&QM&IS) 2018, pp. 9–12. DOI: 0.1109/ITMQIS.2018.8524906.

[14] R.Y. Ivanyushkin, E.A. Saksonov, Y.L. Leokhin, V.A. Netes, M.A. Bykhovsky, “Analysis of hierarchical structure of the corporate network”, *Quality Management, Transport and Information Security, Information Technologies,* Proceedings of the 2017 International Conference on, (IT&QM&IS) 2017.