# Graph model of Fog Computing system

Andriy V. Ryabko[a], Oksana V. Zaika[a], Roman P. Kukharchuk[a] and Tetiana A. Vakaliuk[b,c]

[a]Olexander Dovzhenko Glukhiv National Pedagogical University, 24 Kyievo-Moskovska Str., Glukhiv, 41400, Ukraine
[b]Zhytomyr Polytechnic State University, 103 Chudnivsyka Str., Zhytomyr, 10005, Ukraine
[c]Institute of Information Technologies and Learning Tools of the NAES of Ukraine, 9 M. Berlynskoho Str., Kyiv, 04060, Ukraine

**Abstract**

The development and effective application of Fog Computing technologies require the most complex tasks related to the management and processing of huge data sets, including the tasks of rational construction of low-level networks that ensure the functioning of end devices within the IoT concept. The article describes the use of graph theory methods to solve such problems. The proposed graph model can provide the ability to determine the basic properties of systems, networks, and network devices within the concept of Fog Computing, the optimal characteristics, and ways to maintain them in working condition. This paper shows how to plot graphs, and then customize the display to add labels or highlighting to the graph nodes and edges of pseudo-random task graphs which can be used for evaluating Mobile Cloud, Fog and Edge computing systems. The graphs are described and visualized in Matlab code. Each task has an amount of computational work to perform, expressed in Megacycles per second. Each edge has an amount of data to transfer between tasks, expressed in kilobits or kilobytes of data. The set can be used by researchers to evaluate cloud/fog/edge computing systems and computational offloading algorithms. The task graphs can be used in single-user systems, where one mobile device accesses a remote server, or in multi user systems, where many users access a remote server through a wireless channel.

**Keywords**

fogging computing, multi-level graph model, Internet of Things, Fog Computing, reference architecture OpenFog, graph theory



## 1. Introduction

Nowadays, at the same time as the rapid development of industrial and built electronics will lead to the fact that traditional equipment used in production processes and at home is more than more provided intellectual functions and objects in the network that requires virtually

*QuaInT 2021: Workshop on the Quantum Information Technologies, April 11, 2021, Zhytomyr, Ukraine*
*doors 2021: Edge Computing Workshop, April 11, 2021, Zhytomyr, Ukraine*
ryabko@meta.ua (A.V. Ryabko); ksuwazaika@gmail.com (O.V. Zaika); kyxap4yk1@ukr.net (R.P. Kukharchuk); tetianavakaliuk@gmail.com (Tetiana A. Vakaliuk)
http://pfm.gnpu.edu.ua/index.php/struktura1/2015-04-01-14-50-26 (A.V. Ryabko);
http://pfm.gnpu.edu.ua/index.php/struktura1/2015-04-01-14-50-26 (O.V. Zaika);
http://pfm.gnpu.edu.ua/index.php/struktura1/2015-04-01-14-50-26 (R.P. Kukharchuk);
https://sites.google.com/view/neota/profile-vakaliuk-t (Tetiana A. Vakaliuk)
0000-0001-7728-6498 (A.V. Ryabko); 0000-0002-8479-9408 (O.V. Zaika); 0000-0002-7588-7406 (R.P. Kukharchuk); 0000-0001-6825-4697 (Tetiana A. Vakaliuk)

continuous data processing, growing and requiring computing power of this equipment. The need to control and manage individual mechanisms and machines, as well as the environment, transport flows, production, business, health and education, security, social processes has led to the creation of a large number of devices that interact with people, central data messages and between himself. For their effective functioning, it was necessary to create a global communication system, the quality of the natural state by visiting the Internet, the need to solve management and control problems to create a network concept focused on connecting devices and received from the Internet of Things, IoT.

The term Internet of Things was first used in 1999 by Kevin Ashton, head of the Massachusetts Institute of Technology's Auto-ID Research Center, who suggested that Procter & Gamble use radio frequency tags (RFID) on products to improve supply chain management [1]. Later, this name became commonplace, although its meaning has now expanded significantly.

It should be noted that within the framework of the IoT concept, the basis for the effective operation of both end devices with intelligent functions and the networks formed by them is a real-time operation, reliability, and security, which can not always be provided using a client-server interaction scheme, which is typical of the classic Internet and is based on cloud technology and computing (Cloud Computing) [2].

The concept of Fog Computing is designed primarily to bring data processing and storage closer to the devices that generate and use them. A modified concept of cloud technology applicable to IoT, called Fog Computing (FC), was proposed by Cisco researchers. According to the concept, the FC architecture is three-tiered. At the lowest level – the earth – are billions of things, at the top – many cloud data centers that provide resources for applications that require significant computing power and/or significant amounts of data. And, accordingly, between them is Fog – tens of thousands of geographically distributed smaller control centers, sufficient to solve local problems.

As mentioned above, the concept of Fog computing, in contrast to cloud computing, involves the data processing to the end devices of networks – computers, mobile devices, sensors, and more. It should be noted that now the term Edge Computing has become widespread, which in essence is quite close to Fog Computing, as it also involves the implementation of key operations to collect data processing outside the cloud. However, the key difference between Fog Computing and Edge Computing is considered to be the degree of convergence of data processing points to the edge devices of networks. If the concept of Fog involves sending data generated by end devices for processing and/or storage in local processing centers (Fog Nodes), then in the concept of Edge Computing the main tasks of data processing are solved directly on end devices.

Considering the differences between Cloud Computing and Fog Computing, it should be noted that from the authors' point of view, despite the differences, the contrast between the Fog/Edge Computing models and the Cloud Computing model is erroneous and should not be considered as alternatives but complementary. The basic idea of the Fog Computing concept is to ensure efficient, reliable, and secure interaction of a huge number of devices with each other, with local data centers, and with cloud data centers.

Fog Computing is characterized by the use by users of service functions of resources located on peripheral devices and in the distributed network. The data is located on the client nodes where they should be processed or nearby. The main method of data collection and transmis-

sion is wireless communication. The advantage of networks built according to the concept of Fog Computing over cloud systems is to reduce the latency of the response to the collected data by processing at the place of collection, and for real-time systems, this is one of the key factors. Also, in most cases, the security of Fog Computing systems is higher than for Cloud Computing systems. Another advantage of Fog Computing is the reduction in the amount of data transmitted to the cloud, which reduces network bandwidth requirements, increases data processing speed, and reduces decision delays. Thus, the use of Fog Computing allows you to completely or partially solve some of the most common problems, including:

- large network delays,
- difficulties associated with the mobility of endpoints,
- reliability of communication,
- the high cost of bandwidth;
- unpredictable network congestion,
- large geographical distribution of systems and customers.

OpenFog Consortium, founded in 2015, has proposed a specification of the OpenFog reference architecture, a universal technology model for projects primarily in the field of the IoT, mobile networks, and AI applications. The key members of the consortium are Cisco, Intel, ARM, Dell, Microsoft. According to the proposed model, OpenFog-infrastructure is a set of nodes (Fog Nodes) based on network smart devices that perform data processing, the specification also contains descriptions of options for hierarchical node construction, system deployment models, and examples of possible implementations.

The OpenFog reference architecture is based on the following eight technological principles (criteria):

- security,
- scalability,
- openness,
- autonomy,
- RAS (reliability, availability, suitability for service);
- adaptability;
- hierarchical principle of construction of input elements;
- programmability.

According to Cisco developers, the concept of Fog Computing is best suited for working with machine-to-machine (M2M) systems and devices that use a human-machine interface (HMI). They distinguish three main groups of such devices:

- Data acquisition devices are generated in series by different sensors with a frequency of a few milliseconds to fractions of a second. Examples are devices of security systems and control systems of industrial facilities. They are characterized by low latency requirements for data acquisition and high performance to calculate the required characteristics in real-time.
- Systems responsible for data processing, including operating. Here the input data comes with a frequency of a few seconds to a few minutes. Examples of such systems are devices for visualization of physical processes, technological industrial systems. The requirements for the latency of the collected data and their processing are not as high as for the devices of the first group, but all data are processed in real-time.
- Devices for the collection and processing of historical data, collected at a frequency of several minutes to several days. Example – visualization and reporting systems.

## 2. Theoretical background

Shanhe Yi, Zijiang Hao, Zhengrui Qin and Qun Li point to unresolved cloud computing issues such as unreliability, latency, lack of mobility support, and location awareness [3]. Fog computing can solve these problems by providing resources and services to end-users at the network boundary, while cloud computing is more about providing resources distributed on the core network. Fog provides IoT data processing and storage locally on IoT devices, instead of sending it to the cloud [4].

Case studies Weisong Shi and Schahram Dustdar [5], Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li and Lanyu Xu [5], Nasir Abbas, Yan Zhang, Amir Taherkordi and Tor Skeie [6], Mahadev Satyanarayanan [7] provide a detailed description, definition, and capabilities of Edge-Computing, from cloud technology unloading to smart home and city, mobile networks.

Luis Miguel Vaquero and Luis Rodero-Merino [8] define Fog, considering a variety of technologies such as cloud, sensor networks, peer-to-peer networks, network virtualization functions, or configuration management techniques [8]. Redowan Mahmud, Ramamohanarao Kotagiri and Rajkumar Buyya note that Fog computing is located closer to IoT devices/sensors and expands the capabilities of cloud-based computing, storage, and networking technologies [9]. Ivan Stojmenovic, Sheng Wen, Xinyi Huang and Hao Luan point to Smart Grid technologies, smart traffic lights, software-defined networks [10].

Flavio G. Bonomi, Rodolfo A. Milito, Jiang Zhu and Sateesh K. Addepalli indicate the main characteristics of Fog: a) low latency and location awareness; b) wide geographical distribution; c) mobility; d) a very large number of nodes; e) the predominant role of wireless access; f) the strong presence of streaming and real-time programs; g) heterogeneity [11]. These features make Fog an appropriate platform for several critical IoT services and applications, namely, automotive, smart grids, smart cities, and wireless sensors and network devices (WSANs) in general.

The results of studies by Subhadeep Sarkar and Sudip Misra show that for the scenario where 25% of IoT applications will run with low real-time latency, the average energy consumption for Fog calculations will be 40.48% lower, than in the usual model of cloud computing [12].

Expanding the concepts of Fog Computing is the driving force behind the introduction of Industry 4.0. The development of algorithms and optimization methods is complicated by the complexity of such systems and the lack of real data on Fog systems, which leads to the use of algorithms that are not adapted to real scenarios. Graph-based system parameters allow you to scale and design more realistic test scenarios for future optimization attempts, as well as to determine the features of Fog systems compared to other types of networks [13].

Graph theory is used to construct a load balancing algorithm for Fog network computations [14] based on a dynamic graph distribution [15] that the Fog Cloud Atomization computing system can flexibly build a system network, and the dynamic load balancing mechanism can efficiently configure system resources as well as reduce the consumption of node migration caused by system changes.

Some authors propose a Fog network caching scheme based on the Steiner tree, in which Fog servers, by caching resources, first create a Steiner tree in graphs to minimize the total path weight (or cost) so that the cost of resource caching with this tree can be minimized [3, 16].

Xu Chen and Junshan Zhang propose a hybrid HyFog system for unloading tasks in Fog Computing based on a three-level graph for efficient distribution of tasks between devices. The problem of minimizing the total cost of the task is reformulated as the problem of the minimum weight ratio in the constructed three-level graph, which can be effectively solved using the Blossom Edmonds algorithm [17].

Dmitry Korzun, Aleksey Varfolomeyev, Anton Shabaev and Vladimir Kuznetsov consider two emerging IoT-enabled paradigms: Edge-centric Computing and Fog Computing. They are elaborating their potential for development of smart applications with focus on the dependability and using a mobile assistant for e-tourism as a reference application. They analyze possible concept elements for smart application development and provide recommendations in respect to the dependability [18].

Isaac Lera, Carlos Guerrero and Carlos Juiz propose a fog computing simulator for analyzing the design and deployment of applications through customized and dynamical strategies. They model the relationships among deployed applications, network connections, and infrastructure characteristics through complex network theory, enabling the integration of topological measures in dynamic and customizable strategies, such as the placement of application modules, workload location, and path routing and scheduling of services [19].

Ted H. Szymanski presents describes 300 task graphs which can be used for evaluating mobile cloud, fog and edge computing systems.The task graphs are organized as 3 sets of 100 graphs. Each graph in the first set has the same topology, with $N = 9$ tasks and 6 offloadable tasks. Each graph in the second set has the same topology, with $N = 29$ tasks and 20 offloadable tasks. Each graph in the third set has the same topology, with $N = 23$ tasks and 19 offloadable tasks. Users can also change the number of offloadable components per task graph, in which case the total number of task graphs specified in this paper exceeds 5,000, providing a good basis for the evaluating cloud computing systems [20].

## 3. Research methods

When modeling the operation of both individual elements and the system, designed and/or built on the concept of fog computing as a whole, for example, to assess efficiency, performance, bandwidth, performance, equipment reliability, the delay time for certain types of service, data, software, the system should be presented in the form of several interconnected levels:

- equipment,
- interfaces,
- transport system (network),
- operating systems (OS),
- data,
- services,
- applications.

Such a representation of the object of modeling allows you to use the apparatus of graph theory.

Figure 1 in the form of an oriented graph $G = (Z, L)$ presents a multilevel graph model of the system built according to the concept of Fog Computing. The set of vertices of the graph $Z_{Equip}$, $Z_{ConDev}$, ..., $Z_{ComDev}$, ...; $Z_{Interf}$, $Z_{HMI}$, ..., $Z_{M2M}$, ... .; $Z_{TS}$, $Z_{NetDev}$, ..., $Z_L$, ... ; $Z_{os}$, $Z_{os1}$, ..., $Z_{osn}$, $Z_{Data}$, $Z_{OI}$, ..., $Z_{HD}$, ..., $Z_{BD}$, ...; $Z_S$, $Z_{S1}$, ..., $Z_{Sm}$; $Z_{App}$, $Z_{App1}$, ..., $Z_{Appk}$ are tasks that are solved at each specific level.

The set of arcs contains: $X_{pr}$ – the set of parameters needed to solve these problems and the set of information links – $H$. The set of parameters $X_{pr}$ consists of $H_{pr}^{in}$ – input and $H_{pr}^{out}$ – output parameters.

$$X_{pr} = \{H_{pr}^{in}, H_{pr}^{out}\}$$

with $L = X_{pr} \bigcup H$ and $X_{pr} \bigcap H = \emptyset$.

It should be noted that the output parameters of some tasks (vertices) may be input for others. Information connections are converted into a Boolean matrix $n \bullet n$:

$$H = \|h_{gl}\|_{nxn}.$$

The matrix element $h_{gl}$ characterizes the presence of information connections of problems $g$ and $l$:

$$h_{gl} = \begin{cases} 1 & \text{if the tasks are related by parameters,} \\ 0 & \text{otherwise.} \end{cases}$$

The solution of any problem can be represented as

$$Z_i : \{X_{pr}^{in}\} \Rightarrow Z_{pr}^{out}.$$

It should be noted that there are many mathematical models for solving the problems that correspond to the vertices of the graph $G = (Z, L)$, in the future, they are used to estimate the parameters and characteristics of Fog Computing elements. Each vertex of the graph corresponds to one or more nodes, and the output parameters of some nodes can be input to others. Michaela Iorga, Larry Feldman, Robert Barton, Michael J. Martin, Nedim Goren and Charif Mahmoudi note that Fog nodes are either physical components (e.g., gateways, switches, routers, servers, etc.) or virtual components (e.g., virtual switches, virtual machines, clouds, etc.) that are closely related to intelligent endpoints, devices or access networks, and provide computing resources for these devices [21].

Each parameter for solving problems is characterized by a vector of characteristics of parameters, which includes: units of measurement of the parameter; the vertex in which this parameter is the source; the vertex in which the parameter is input, the level number, and other individual characteristics. The units of measurement of parameters are determined by the International System of Units (SI) and by the semantics of specific modeling tasks. In this case, all parameters should be divided into two groups: 1) parameters corresponding to the SI system; 2) all other parameters.

For the same parameters, the units of measurement may be different; therefore, it is necessary to perform parameter matching. As mentioned above, the OpenFog reference architecture is based on the following eight criteria:

$$K = \{K_1^u, ..., K_8^u\},$$

where $K$ is a compound criterion; $u$ is the level number ($u = 1, ..., 7$); $K_i^u$ – criterion of a specific level, for example, security at the level of services.

This method can be used in this study, because using the graph $G = (Z, L)$, it is possible to integrate models for calculating and evaluating various parameters depending on specific tasks, to carry out multivariate calculations, and to effectively evaluate almost all parameters and characteristics of Fog Computing elements. The graph model clearly shows the mutual influence and interrelation of tasks and parameters, and modeling can be started with any task that corresponds to the top of the graph (one or more).

## 4. Results

Let us dwell in more detail on the results of modeling the graph of a system designed and/or built on the concept of Fog Computing (figure 1).

1st level – equipment. The vertices of the graph $Z_{Equip}$, $Z_{ConDev}$, ..., $Z_{ComDev}$, ... correspond to this level. The vertex of $Z_{Equip}$ defines all problems which are solved at this level; vertices of $Z_{ConDev}$, ... – (connecting devices) tasks that are solved at the level of a wide range of devices connected to the network; vertices $Z_{ComDev}$, ... – tasks for devices that allow you to perform the necessary calculations. An example of the tasks that are solved at this level is the calculation of the reliability and performance of devices.

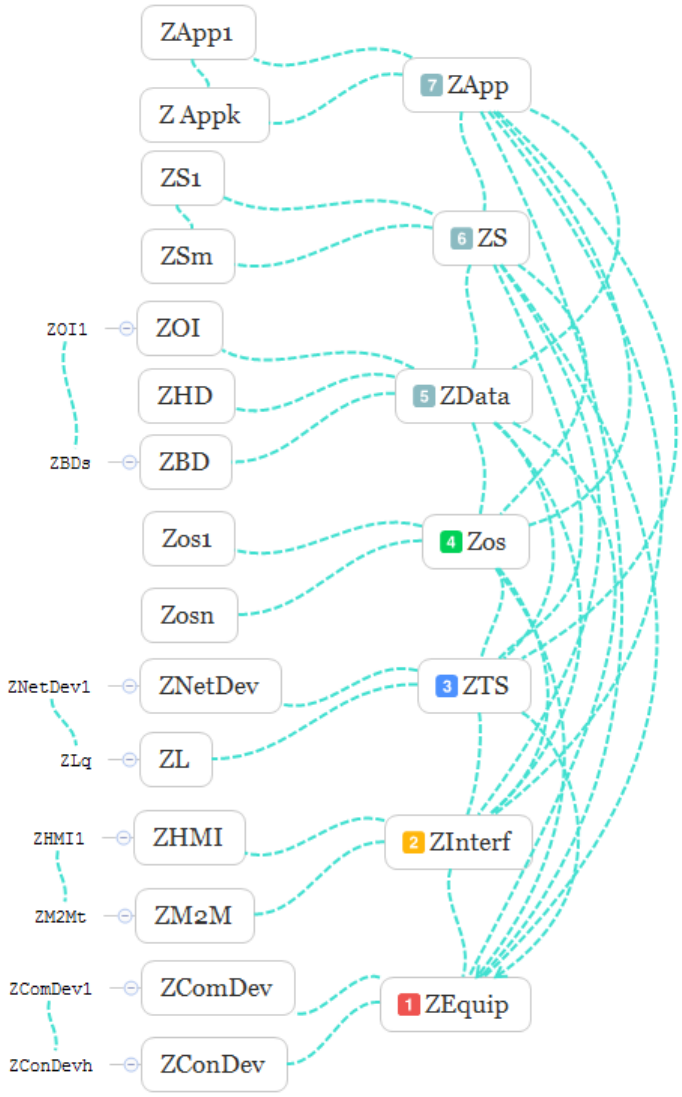


**Figure 1:** Graph model of the system, which is implemented according to the concept of Fog Computing

2nd level – interfaces. As mentioned above, Fog calculations are best suited for working with inter-machine interaction systems – M2M, and devices that use a human-machine interface – HMI. The interface level corresponds to the vertices of the graph $Z_{Interf}$, $Z_{HMI}$, ..., $Z_{M2M}$, .... The vertex of the $Z_{Interf}$ describes the tasks specific to this level; vertices $Z_{HMI}$,

..., $Z_{M2M}$, ... – define tasks for systems with inter-machine and human-machine interface, respectively. M2M – machine-machine interaction (Machine-to-Machine, Mobile-to-Machine, Machine-to-Mobile) – the name of the technology (sum of technologies), which allows data transfer between different devices, and it can be groups of devices, such as public transport. Work on M2M is coordinated by the following organizations: the Eclipse Foundation, the Focus Groupon Machine-to-Machine, a member of the International Telecommunication Union, and the TR-50 M2M Intelligent Devices Engineering Committee.

Types of M2M: stationary M2M, such as process control, payment terminals, meters, etc., and mobile M2M, for example, for fleet management, where M2M is used as an on-board device for monitoring, diagnostics, navigation, security, and mobile communications. M2M applications include access systems, premises security systems, security systems, remote control and management of equipment, transport and monitoring of moving objects, vending machines, payment terminals, healthcare, etc. HMI – human-machine interface – a concept that encompasses engineering solutions that provide human interaction with controlled objects (machines, systems, devices).

The HMI can be a computer, standard software, a simple remote control with a set of LED indicators, and so on. Modern computers are focused on streaming architecture, the implementation of intelligent human-machine interface, which provides not only a systematic solution but also the ability of the machine to logical thinking and self-learning, to associative information processing and drawing logical conclusions. The requirements that different users impose on the HMI vary widely. The implementation of an intelligent human-machine interface is associated with the ability to solve problems of recognition and understanding of natural language, for this, there are recognition systems (language, handwritten texts, images). Creating a user-friendly and efficient human-machine interface is an urgent task. Also among the tasks of the interface level can be distinguished, for example, the choice of standard interface buses; reliability assessment at the interface level, and others.

3rd level – transport system (network). The transport system is used to transmit information and contains nodes of the Fog infrastructure – switches, routers, etc. The vertex $Z_{TS}$ of the graph $G = (Z, L)$ defines the general tasks characteristic of the 3rd level, and the vertices $Z_{NetDev}$, ... – describe the tasks that are solved at the level of network devices; vertices $Z_L$, ..., $Z_{Lq}$, ... – tasks that are solved at the level of communication channels. As the tasks are solved at this level it is possible to result in the following – a choice of a communication channel; channel bandwidth estimation; calculation of the delay factor of network equipment; estimation of message delay and many others.

4th level – operating systems (OS). Here you should consider the presence of different types of operating systems (UNIX-like OS, Windows, macOS, etc.). The vertex $Z_{os}$ of the graph describes the general tasks characteristic of the OS level.

Vertices $Z_{os1}$, ..., $Z_{osn}$ are tasks that are solved for each specific operating system, for example:

- calculation of the coefficient of relative losses of OS performance for a multiprocessor system,
- determining the average processing time of the OS request,
- estimation of the average time spent on access to external memory and analysis of the

intensity of OS requests to external memory devices,

- assessment of the reaction time of the OS in solving specific problems,
- an estimate of the average time required to transmit the OS request,
- estimate the time of access to RAM,
- optimization of the core structure of open OS by the criterion of information security,
- estimation of time of detection of errors in processes,
- calculation of the probability of skipping the controlled signal (quantitative characteristics for the tasks of monitoring the integrity of OS files) and many others.

5th level – data. The vertices $Z_{Data}$, $Z_{OI}$, ...., $Z_{HD}$, ..., $Z_{BD}$, ... of the graph $G = (Z, L)$ correspond to this level. The $Z_{Data}$ vertex describes general tasks, $Z_{OI}$, ...., $Z_{HD}$, ..., $Z_{BD}$, ... vertices – tasks specific to operational information (real-time analysis), historical data (transaction analysis), and long-term storage (BigData analysis).

Examples of tasks to be solved at this level:

- prognostic calculation of the speed of new data generation,
- optimization of file placement and processing of requests to the database,
- estimation of data volume,
- data compression,
- distributed calculations when planning requests to the database,
- assessment of the integrity of information at the level of links and other tasks.

6th level – services. These can be various services, such as online services (like Uber), streaming services (like Netflix, Amazon Prime, Hulu, and Crunchyroll), etc.

The level of services corresponds to the vertices $Z_S$, $Z_{S1}$, ..., $Z_{Sm}$ graph model. Vertex $Z_S$ defines general tasks for service level, vertices $Z_{S1}$, ..., $Z_{Sm}$ – tasks for different types of service.

Examples of tasks:

- calculation of productivity for the 6th level,
- assessment of service quality in virtual VPN channels,
- optimization of system services by network resources,
- assessment of the security of transmission of confidential information in broadcast communication channels,
- maximum support for different types of 6th level traffic, etc.

7th level – applications. Applications are research software, computer-aided design systems, games, applications for artists, geographically distributed applications for pipeline monitoring, smart devices in the car, SmartGrid, traffic light control systems, etc.

The vertices $Z_{App}$, $Z_{App1}$, ..., $Z_{Appk}$ of the graph $G = (Z, L)$ correspond to this level. Vertex $Z_{App}$ describes the general tasks of the application level, the vertices of $Z_{App1}$, ..., $Z_{Appk}$ – tasks for different types of applications. These can be applications installed on computers, tablets, smartphones of users.

Some examples of tasks:

- calculation of maximum productivity for the 7th level,
- distribution of application tasks between users according to the criterion of weighted average route length,
- prognostic estimate of the conditional average service time of the application required to perform the task lasting in $nt$ period,
- estimation of the average time of the decision of applied problems,
- calculation of exchange time with external memory in the process of solving applied corporate tasks,
- prognostic calculation of the time required for data processing in the application system,
- calculation of the average service time of the application for algorithms of non-priority service disciplines,
- scalability for the 7th level.

The main properties of the multilevel graph model of Fog Computing system are:

1. integration – the ability to solve individual (partial) problems depending on the specific situation,
2. universality,
3. adequacy,
4. accuracy,
5. efficiency,
6. property of development – the model is created and functions taking into account additions, improvements, and updates.

The requirements of a high degree of universality, accuracy, adequacy on the one hand, and its high efficiency, on the other hand, are contradictory.

In the process of analyzing the graph model of the system, which is implemented by the concept of Fog Computing, the graph $G = (Z, L)$ is decomposed into typical subgraphs (the necessary subgraphs are allocated to save time and computing resources), the parameters of the corresponding vertices are determined by set of connections between vertices (tasks). Then there is a selection of appropriate models (formulas) to calculate the necessary parameters for solving specific problems. Next, you need to analyze the entire graph model. Using graph

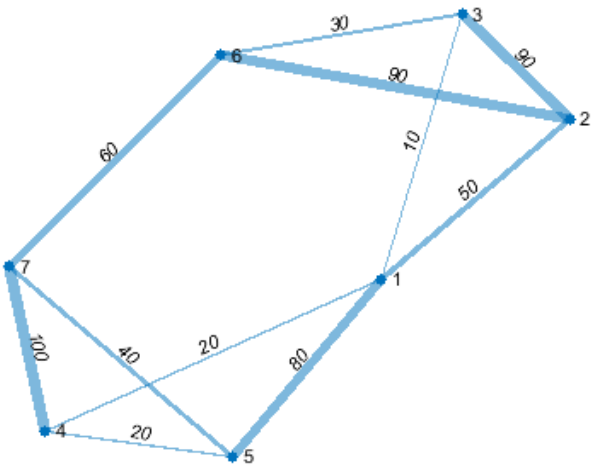


**Figure 2:** Graph with the width of the edges, which is proportional to their weight

$G = (Z, L)$ it is possible to integrate models for calculation and estimation of various parameters depending on concrete tasks, to carry out various calculations, to estimate practically all parameters and characteristics of Fog Computing system elements. The graph model clearly shows the mutual influence and relationship of tasks and parameters, and modeling can begin with any problem that corresponds to the vertex of the graph (one or more).

Fog Computing network simulations were performed in Matlab. Since the 2015 release, Matlab has been able to work with graphs using the $G = graph()$ function. After you create a graph that simulates a network, you can get information about the graph by using object functions to perform object queries. For example, you can add or remove nodes or edges, define the shortest path between two nodes, or find a specific graph node:

```
G = graph([1 1], [2 3]);
e = G.Edges
G = addedge(G,2,3)
G = addnode(G,4)
plot(G)
```

The next step in describing the relationships between objects using graphs is to give the edges certain symbolic values, qualitative characteristics, called weights. In the simplest cases, this may be the ordinal numbering of the edges, which is checked by the order of their consideration (priority or hierarchy). Rib weight can mean length (message paths), bandwidth (communication lines), load. Weight can be attributed not only to the ribs but also to the vertices. For example, the vertices that correspond to the Fog nodes of the network can characterize the number, bandwidth, and so on. Next, a graph is constructed, indicating the weight of the edges and making the width of the edges proportional to their weight (figure 2).

We used a set of pseudo-random task graphs which can be used for evaluating Mobile Cloud, Fog and Edge Computing systems. The pseudo-random task graphs are based upon graphs that

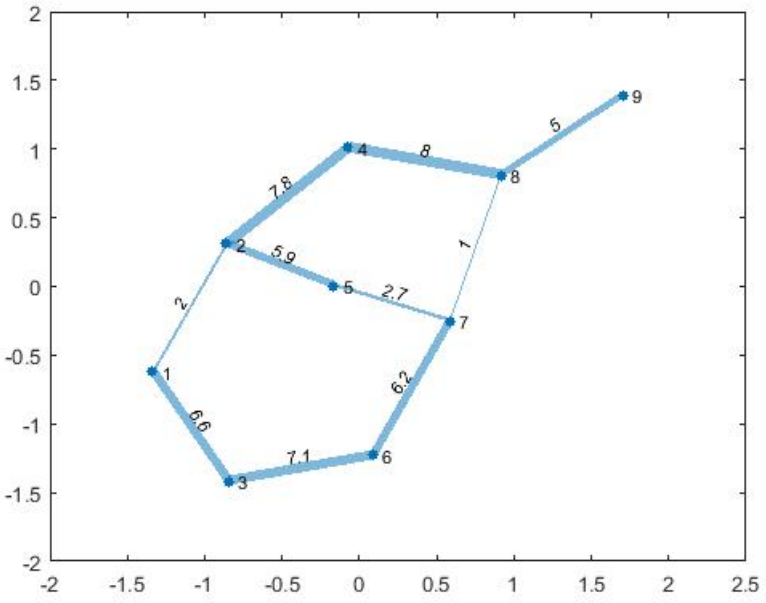


**Figure 3:** Three task graphs each with N = 9 nodes and 6 offloadable tasks. Tasks 1,6 and 9 execute locally. Edge weights are in Kilobits

have previously developed by Ted H. Szymanski [20]. Researchers can download these 300 task graphs and analyse their own configurations of mobile cloud, fog and edge computing systems with different mobile device parameters and cloud/fog/edge server parameters, but the code does not provide the ability to visualize graphs and properties of nodes and edges.

For example, the first set of graphs is based upon a task graph presented in [20]. Reference [20] provided the 3 task graphs shown in figure 3, and it provided the optimal energy for each graph (under certain assumptions). In figure 3, the task graphs consists of 9 tasks and 10 edges. Tasks 1, 6 and 9 must execute locally, and the remaining tasks can execute locally or remotely. The task complexities are expressed in Mega-cycles per task, and the edges are annotated with the data size expressed in kilobits. The task graph in Fig. 3 contains only in one variable, where the complexity of task 8 changes from 3 to 20 Mega-cycles.

Graph plots are the primary way to visualize graphs and networks created using the graph and digraph functions. After you create a GraphPlot object, you can modify aspects of the plot by changing its property values. This is particularly useful for modifying the display of the graph nodes or edges. GraphPlot properties control the appearance and behavior of plotted graphs. By changing property values, you can modify aspects of the graph display. Use dot notation to refer to a particular object and property. Let's plot a graph by marking the edges with their weights and making the widths of the edges proportional to their weights. Use a scaled version of the edge weights to define the width of each edge so that the widest line has a width of 5. This example shows how to plot graphs:

```
EDGE_src(1,1:10)=[1,1,2,2,3,4,5,6,7,8,];
```

```
EDGE_dst(1,1:10)=[2,3,4,5,6,8,7,7,8,9,];
EDGE_bits(1,1:10)=[200,660,780,590,710,80,270,620,100,500,]
G=graph(EDGE_src,EDGE_dst,EDGE_bits)

G =
graph with properties:
Edges: [10x2 table]
Nodes: [9x0 table]
>> LWidths = 5*G.Edges.Weight/max(G.Edges.Weight);
plot(G,'EdgeLabel',G.Edges.Weight,'LineWidth',LWidths)
```

The second task graph is an extended version of the previous task graph, and the seed graph is shown in figure 3. Three instances of the task graph from set 1 (in figure 2) are placed in parallel, between the entry node 1 and exit node 29. The seed graph has 29 nodes and 36 edges. Nodes 1, 29 and 7 other randomly selected nodes execute locally. Each task graph in set 2 thus has 20 offloadable tasks. This example shows how to plot graphs:

```
>>EDGE_src(1,13:24)=[1,11,11,12,12,13,14,15,16,17,18,19,];
EDGE_dst(1,13:24)=[11,12,13,14,15,16,18,17,17,18,19,29,];
EDGE_bits(1,25:36)=[200,200,660,780,590,710,80,270,620,100,500,300,];
G=graph(EDGE_src,EDGE_dst,EDGE_bits)

G=
Graph with properties:
Edges: [36x2table]
Nodes: [29x0table]
>>LWidths=5*G.Edges.Weight/max(G.Edges.Weight);
plot(G,'EdgeLabel',G.Edges.Weight,'LineWidth',LWidths)
```

The first task graph in set 2 is identical to the one shown in figure 4.

The practical application of the model can, for example, help mobile devices overcome resource constraints by unloading computing tasks on cloud servers. The task of the cloud is to minimize the time of data transfer and execution of tasks to the user, whose location changes due to the mobility and power consumption of the mobile device.

Ensuring satisfactory computational performance is particularly difficult in Fog Computing. The graph model of calculations will allow bringing effectively computing power Fog to the mobile user. The graph model consists of remote cloud nodes and local cloud nodes that are connected to the wireless access infrastructure. Evaluating the effectiveness of our method using experimental modeling in Matlab shows good results, which show that this method allows you to calculate the ability to reduce task execution time and power consumption of mobile devices.

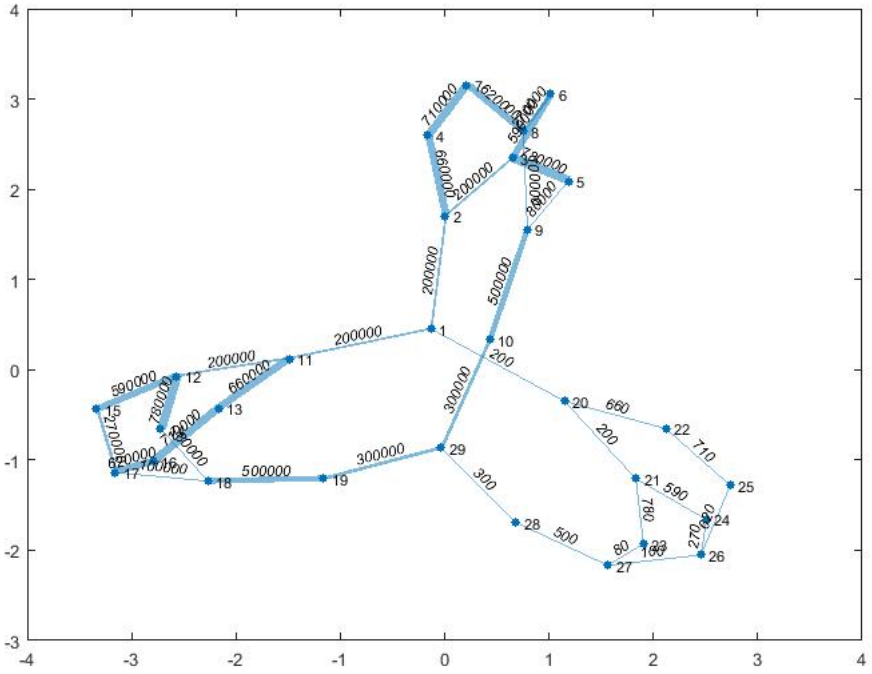


**Figure 4:** Task graph with N = 29 nodes and 20 offloadable tasks based upon previous figure

## 5. Conclusions

The developed graph model can be a very versatile tool for optimizing existing networks and for implementing new systems and networks created by the concept of Fog Computing, and, above all, focused on the IoT. The IoT and Fog computing is just entering everyday life, and the model provides the ability to calculate, perform a prognostic assessment, optimize the necessary characteristics of the entire system and individual elements, identify bottlenecks and redundancy at the design stage taking into account OpenFog reference architecture, criteria (technological principles) and types of data processing systems. The proposed graph model can provide the ability to determine the basic properties of systems, networks, and network devices within the concept of Fog Computing, the optimal characteristics, and ways to maintain them in working condition. A promising task is to create an algorithm for calculating the shortest path between nodes when dynamically changing the weight of the edges of the graph (for example, when changing the location of a mobile device) to solve practical problems of unloading mobile resources.

This paper shows how to plot graphs, and then customize the display to add labels or highlighting to the graph nodes and edges of pseudo-random task graphs which can be used for evaluating Mobile Cloud, Fog and Edge computing systems. The graphs are described and visualized in Matlab code.

We plan to develop a set of use-case scenarios that we analyze to determine the graph based parameters of the system that allows us to scale and generate a more realistic testing scenario for future optimization attempts as well as determine the nature of fog systems in comparison to other networks types.

## References


[1] T. Kramp, R. van Kranenburg, S. Lange, Introduction to the Internet of Things, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 1–10. doi:10.1007/978-3-642-40403-0_1.

[2] O. Markova, S. Semerikov, A. Striuk, H. Shalatska, P. Nechypurenko, V. Tron, Implementation of cloud service models in training of future information technology specialists, CEUR Workshop Proceedings 2433 (2019) 499–515.

[3] S. Yi, Z. Hao, Z. Qin, Q. Li, Fog computing: Platform and applications, in: 2015 Third IEEE workshop on hot topics in web systems and technologies, HotWeb, 2015, pp. 73–78. doi:10.1109/HotWeb.2015.22.

[4] H. Atlam, R. Walters, G. Wills, Fog computing and the internet of things: A review, Big data and cognitive computing 2 (2018) 10. doi:10.3390/bdcc2020010.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of things journal 3 (2016) 637–646. doi:10.1109/JIOT.2016.2579198.

[6] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal 5 (2017) 450–465. doi:10.1109/JIOT.2017.2750180.

[7] M. Satyanarayanan, The emergence of edge computing, Computer 50 (2017) 30–39. doi:10.1109/MC.2017.9.

[8] L. Vaquero, L. Rodero-Merino, Finding your way in the fog: Towards a comprehensive definition of fog computing, ACM SIGCOMM Computer Communication Review 44 (2014) 27–32. doi:10.1145/2677046.2677052.

[9] R. Mahmud, R. Kotagiri, R. Buyya, Fog computing: A taxonomy, survey and future directions, Internet of everything (2018) 103–130. doi:10.1007/978-981-10-5861-5_5.

[10] I. Stojmenovic, S. Wen, X. Huang, H. Luan, An overview of fog computing and its security issues, Concurrency and Computation: Practice and Experience 28 (2016) 2991–3005. doi:10.1002/cpe.3485.

[11] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012, pp. 13–16. doi:10.1145/2342509.2342513.

[12] S. Sarkar, S. Misra, Theoretical modelling of fog computing: a green computing paradigm to support iot applications, IET Networks 5 (2016) 23–29. doi:10.1049/iet-net.2015.0034.

[13] N. Verba, K. Chao, A. James, J. Lewandowski, X. Fei, C. Tsai, Graph analysis of fog computing systems for industry 4.0, in: 2017 IEEE 14th International Conference on e-Business Engineering (ICEBE), 2017, pp. 46–53. doi:10.1109/ICEBE.2017.17.

[14] I. Lera, C. Guerrero, C. Juiz, Availability-aware service placement policy in fog computing

based on graph partitions, IEEE Internet of Things Journal 6 (2018) 3641–3651. doi:10.1109/JIOT.2018.2889511.

[15] S. Ningning, G. Chao, A. Xingshuo, Z. Qiang, Fog computing dynamic load balancing mechanism based on graph repartitioning, China Communications 13 (2016) 156–164. doi:10.1109/CC.2016.7445510.

[16] S. Yi, C. Li, Q. Li, A survey of fog computing: concepts, applications and issues, in: Proceedings of the 2015 workshop on mobile big data, 2015, pp. 37–42. doi:10.1145/2757384.2757397.

[17] X. Chen, J. Zhang, When d2d meets cloud: Hybrid mobile task offloading in fog computing, in: 2017 IEEE international conference on communications (ICC), 2017, pp. 1–6. doi:10.1109/ICC.2017.7996590.

[18] D. Korzun, A. Varfolomeyev, A. Shabaev, V. Kuznetsov, On dependability of smart applications within edge-centric and fog computing paradigms, in: 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2018, pp. 502–507. doi:10.1109/DESSERT.2018.8409185.

[19] I. Lera, C. Guerrero, C. Juiz, Yafs: A simulator for iot scenarios in fog computing, IEEE Access 7 (2019) 91745–91758. doi:10.1109/ACCESS.2019.2927895.

[20] T. H. Szymanski, 300 pseudo-random task graphs for evaluating mobile cloud, fog and edge computing systems, 2018. doi:10.21227/kak5-8n96.

[21] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, C. Mahmoudi, Fog computing conceptual model, Natl. Inst. Stand. Technol. Spec. (2018) 1–15. doi:10.6028/NIST.SP.500-325.