future internet

Article

DD-FoG: Intelligent Distributed Dynamic FoG Computing Framework

Volkov Artem [1], Kovalenko Vadim [1], Ibrahim A. Elgendy [2], Ammar Muthanna [1,3,*] and Andrey Koucheryavy [1]

[1] Department of Telecommunication Networks and Data Transmission, The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, 193232 Saint Petersburg, Russia; artemanv.work@gmail.com (V.A.); vadimkovalenko1996@gmail.com (K.V.); akouch@mail.ru (A.K.)
[2] Department of Computer Science, Faculty of Computers and Information, Menoufia University, Shibin el Kom 32511, Egypt, ibrahim.elgendy@ci.menofia.edu.eg
[3] Department of Computer Science, RUDN University, Peoples' Friendship University of Russia, 6 Miklukho-Maklaya Str., 117198 Moscow, Russia
* Correspondence: muthanna.asa@spbgut.ru

Abstract: Nowadays, 5G networks are emerged and designed to integrate all the achievements of mobile and fixed communication networks, in which it can provide ultra-high data speeds and enable a broad range of new services with new cloud computing structures such as fog and edge. In spite of this, the complex nature of the system, especially with the varying network conditions, variety of possible mechanisms, hardware, and protocols, makes communication between these technologies challenging. To this end, in this paper, we proposed a new distributed and fog (DD-fog) framework for software development, in which fog and mobile edge computing (MEC) technologies and microservices approach are jointly considered. More specifically, based on the computational and network capabilities, this framework provides a microservices migration between fog structures and elements, in which user query statistics in each of the fog structures are considered. In addition, a new modern solution was proposed for IoT-based application development and deployment, which provides new time constraint services like a tactile internet, autonomous vehicles, etc. Moreover, to maintain quality service delivery services, two different algorithms have been developed to pick load points in the search mechanism for congestion of users and find the fog migration node. Finally, simulation results proved that the proposed framework could reduce the execution time of the microservice function by up to 70% by deploying the rational allocation of resources reasonably.

Keywords: 5G-/IMT-2020; IMT-2030; framework; mobile edge computing; fog computing

check for updates

Citation: Artem, V.; Vadim, K.; Elgendy, I.A.; Muthanna, A.; Koucheryavy, A. DD-FoG: Intelligent Distributed Dynamic FoG Computing Framework. *Future Internet* **2022**, *14*, 13. https://doi.org/10.3390/fi14010013

Academic Editor: Michael Sheng

Received: 28 October 2021
Accepted: 24 December 2021
Published: 27 December 2021

1. Introduction

5G/IMT-2020 infrastructure based on several new architecture solutions can provide new technical opportunities for realizing new services with high data transfer speed and low latency such as tactile Internet, medical services, augmented reality applications and others [1–3]. However, as we know, tactile Internet applications require 1 ms RTT and thereby, modern approaches confront light speed limitations in the optical physical medium, which ultimately restrict the service distance. As result, researchers have introduced two solutions to tackle this issue. The first solution is based on changing the physical technology with quantum communication, network and computational resources decentralization. Whereas, the second solution is based on applying the compute allocation to improve Quality of Service levels through the integration of fog and MEC with the microservices support in which dynamicality of devices is considered. This paper focused on the second solution[4,5].

According to several International Telecommunication Union recommendations, such as the ITU-R M.2083-0 IMT vision—"Framework and overall objectives of the future development of IMT-2020 and beyond", IMT-2020 infrastructure will be based on software-defined networking (SDN) and network function virtualization (NFV), including computational technologies like a MEC and fog computing [6].

Currently, few software architecture solutions have been proposed for developing IT solutions including IoT-based applications. One of the modern software architectures is the microservices approach. This architecture provides more opportunities than monolithically and difficult structures [7,8]. Ideally, it's number of independent microservices in each can be worked with others via message broker, application programming interface gateway (api-gw). The comprehensive and difficult solution based on microservices generally has the special microservice/service for monitoring and controlling every other microservices in structure and message traffic between them [9].

In a general view of the fog system, MEC and SDN with NFV infrastructure technologies can provide more positive synergy effect [10–12]. Taking into account the dynamics of user equipment, the new quality challenges, the new ICT infrastructure can ensure the necessary characteristics. An important part of service is the software (platforms, large servers, etc.) and as marked above, the new services with high load should be based on the microservices approach. In this way, microservices are more efficient architectural approaches for implementing fog computing. However, microservices development and deployment are currently challenging in fog and MEC architecture. Therefore, the deployment scheme must be carefully considered. Motivated by such considerations, we proposed a new framework for deploying services based on MEC, fog infrastructure in accordance with user dynamics. The main contributions of this paper are summarized as follows:

- In this paper, we proposed a new framework for deploying services based on MEC, FoG infrastructure in accordance with user dynamics. The key feature of this framework (DD-fog) is the microservices migration on the necessary FoG-node for providing a stable quality level for each service.
- K means-based algorithm is developed and tested for peak loads center searching.
- Swarm-based algorithm is proposed and tested to define the fog-node with necessary resource allocation.
- Simulation results proved that the proposed framework could reduce the execution time of the microservice function due to the rational allocation of resources by up to 70%.

The rest of the paper is presented in several sections: The related works and problem statement are detailed in Section 2. Section 3 introduces the proposed system. Afterward, Section 4 presents the proposed algorithms and Section 5 discusses the framework's algorithms realization. Finally, Section 6 concludes the paper.

2. Related Works

Tran et al. [13] have devoted to the MEC technology, in which the authors have reviewed the advances in the development of MEC networks, as well as the main problems arise when organizing a network architecture with MEC such as security, resource management, support for mobility, resource and load distribution, service availability and interoperability. In addition, it compared the main characteristics of MEC technologies, which allowed for expanding the capabilities of the network, and the cloud radio access network C-RAN, and centralized the functions of the base station through virtualization. Whereas, in [13], three scenarios of interaction in networks with MEC were considered. In the first scenario, in addition to the three levels of the network architecture represented by the end-user level, the level of the radio access network boundary with MEC servers connected to the base stations, the level of the remote cloud server, there is an additional layer between the MEC servers and the remote computing cloud, which ensures the interaction of end users with each other and with MEC and remote cloud servers. Moreover, it is decided where the end-user request will be processed. The second scenario deals with the sharing of the video cache on MEC servers, which means that the same file is located on several servers, but in different encodings. Thus, each server will provide access to the video not only in the proposed, but also in other required formats through links associated with the video. Finally in the third scenario, two-level interference cancellation is considered: neighboring cells are combined into clusters, a coordinated multipoint transmission and reception (CoMP) method is applied.

Furthermore, Li et al. [14] have presented a network architecture using fog computing, consisting of end-user equipment, a fog server responsible for forming a pool of devices, and a remote cloud server, and also proposed a dynamic policy for mobile cloud computing clusters DMCCP. The fog server will organize a cluster of cloudlet mobile devices to reduce the load that comes from the user. Using MRM mobile resource monitoring, the fog server tracked the amount of resources available on each device, the number of device resources, and considered the main indicators taken into account when calculating resources namely CPU health, memory, battery power, network capacity, and a factor that determines the ability of other devices to complete a task. When forming the clusters, a heuristic method was used. Using this method, the fog server first formed a list of available devices in descending order of the resources provided, and also determines the amount of resources R required to complete the task. Then, sequentially, from the total number of resources required to complete the task, it subtracts the number of device resources with the largest characteristics until R is less than 0, after which the formation of the cloudlet begins.

Elsewhere, ref. [15] is devoted to the application of the mechanism of dynamic load balancing of fog computing based on graph redistribution. Papers dealing with load balancing and resource scheduling for cloud computing in Fog networks were studied. The paper considers the hierarchy of the Framework for “foggy” systems, which consists of four levels: the level of physical resources, the level of atomization, the level of service management and the level of platform management. At the level of atomization, a set of resources for cloud atomization is formed from physical resources. The article describes the course of the cloud atomization process for “foggy” computing systems, as well as an algorithm using graphs for the dynamic formation of nodes of virtual machines. To prove the efficiency of the proposed algorithm, the authors conducted a simulation of the operation of a “fog” computing system on real equipment.

Table 1 provides a description of the main problems under consideration and the proposed solutions for each of the related works.

Taking into account the described issues, infrastructure technologies of 5G/IMT-2020 communication networks and the trend of their development in communication networks 2030, it is necessary to form a system solution (framework) that will allow maintaining the required level of quality of service by migrating microservices to fog-node that meets the technical requirements of the migrating microservice. In the proposal, it is necessary to consider the AI algorithms to lay the necessary technological level for future expansion and support increasing needs.

Table 1. Related works.

Existing Works	Problems Addressed	Proposed Solutions
[13]	• The traditional way of moving computing to a remote cloud can lead to high delays. • Mobile video streaming traffic is projected to account for 72 percent of total mobile data traffic by 2019, which could put enormous pressure on network operators.	• In our study, we proposed to use an additional layer between the MEC servers and the remote computing cloud, which will select the level of processing for end-user requests. • In addition, by sharing the video cache on MEC servers, the same file is stored on multiple servers, but in different encoding. • Moreover, a two-tiered interference cancellation infrastructure has been proposed.

Table 1. *Cont.*

Existing Works	Problems Addressed	Proposed Solutions
[14]	Mobile devices cannot always meet the requirements for organizing of high resource consuming applications, and capability of fog servers are still limited due to the high cost of deployment.	• To address this issues, we proposed a dynamic policy for mobile cloud DMCCP clusters. • Besides, we propose a formula for calculating the amount of resources available to the device.
[15]	Due to the high polymerization calculation mode, compute clouds cannot fully utilize the resources of the edge device.	• Fog systems were analyzed from a hierarchical perspective, which are consisting of four levels. • Additionally, in this paper, the cloud atomization process for fog computing systems is presented as well as a graph-based algorithm for the dynamic formation of nodes of virtual machines.

3. Proposed System

At the beginning of 2020, one can already observe the gradual introduction of new technologies of networks and computing infrastructure, as well as new services. At the same time, the concept of IMT-2020 is being replaced by the concept of generation 2030 networks.

In addition to networking (SDN/NFV), approaches to building a computing infrastructure are also replacing, which will eliminate some components of the round-trip delay introduced by the network in the process of transmitting packets with information to the data center. These technologies are MEC (multi-access edge computing) and fog computing. These concepts for organizing cloud computing structures will allow data processing to be moved to the edge of the network and even to distributed Fog structures (in fact, devices).

To achieve a positive synergistic effect from the joint use of the above technologies, the following solution is proposed for combining them (Intelligent Framework of Distributed Dynamic fog Computing System). This solution will allow us to achieve new opportunities for the implementation of custom services based on the Internet of Things with microservices architectural support, within the framework of stringent requirements for them.

3.1. Proposed Framework of Distributed Dynamic Fog Computing System (DD-Fog)

This framework is designed to integrate cloud structures (MEC and fog), a network infrastructure built on the basis of SDN/NFV technologies (Figure 1), taking into account the level of orchestration. Thus, the combined infrastructure of network and computing resources allows implementing new approaches to the distribution of computations, including at the network edge and on the devices themselves. In this case, we consider an approach to management through the northbound infrastructure interface (application layer of the computational resource orchestrator, application layer of the SDN/NFV network controller (s)). This feature allows you to implement the business logic of both network and service operator services. However, in addition, due to the inherent level of abstraction from physical resources, it is possible to create management applications-services that implement the system and business logic of the operator for managing devices of the controlled infrastructure and resources. In this case, as has already been stated in several scientific articles [16–19], as well as documents of international standardizing organizations (SDO's), the path of automation, as well as intellectualization of the network, lies through the development of services, data processing algorithms that belong to the Artificial Intelligence class. Thus, the monitoring and control system, which is an infrastructure application (software complex), which implements in itself algorithms of machine intelligence as algorithms, makes it possible to automate to a certain extent and, moreover, make the infrastructure intelligent, without the need for constant control and management by people , highly qualified specialists. The resolution of this task will reduce such an indicator as OPEX by reducing the staff of qualified network administrators. Furthermore, it will also allow you to qualitatively change the network, in terms of their reaction to actions from the outside (traffic growth, change in the traffic profile)—the network will adjust to current needs, redistribute the load, taking into account the QoS rules.

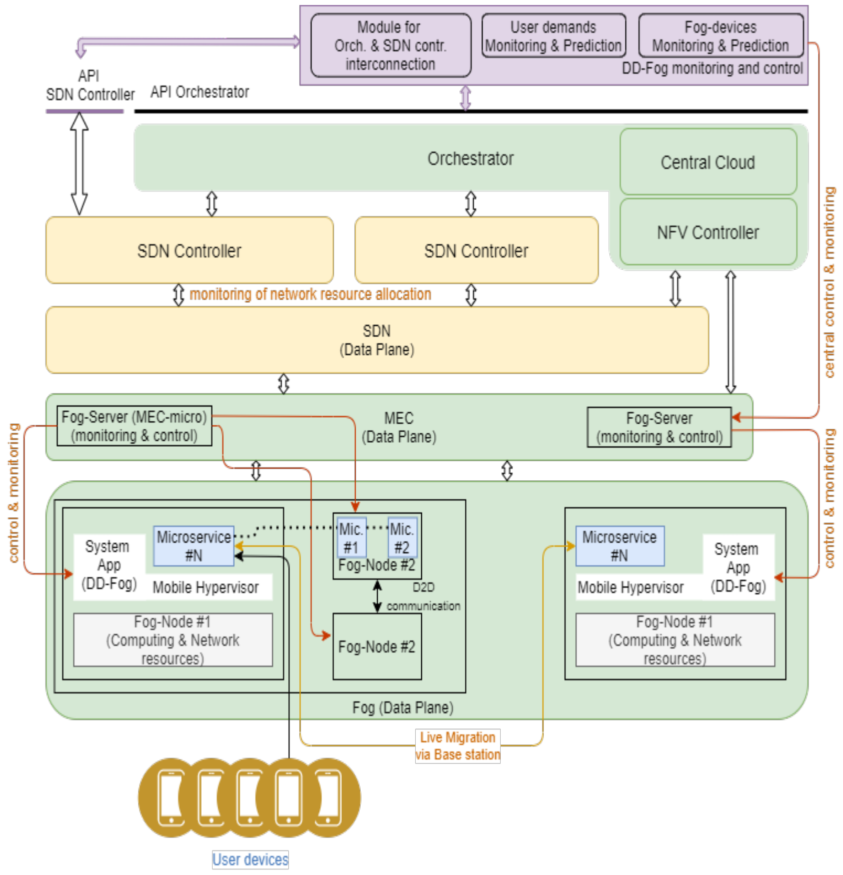
Module for Orch. & SDN contr. interconnection
User demands Monitoring & Prediction
Fog-devices Monitoring & Prediction
DD-Fog monitoring and control
API SDN Controller
API Orchestrator
Orchestrator
Central Cloud
NFV Controller
SDN Controller
SDN Controller
monitoring of network resource allocation
SDN (Data Plane)
central control & monitoring
Fog-Server (MEC-micro) (monitoring & control)
MEC (Data Plane)
Fog-Server (monitoring & control)
control & monitoring
control & monitoring
System App (DD-Fog)
Microservice #N
Mobile Hypervisor
Fog-Node #1 (Computing & Network resources)
Mic. #1
Mic. #2
Fog-Node #2
D2D communication
Fog-Node #2
Microservice #N
System App (DD-Fog)
Mobile Hypervisor
Fog-Node #1 (Computing & Network resources)
Fog (Data Plane)
Live Migration via Base station
User devices

Figure 1. The common architecture of the framework.

In addition, such a network will take into account its power characteristics, monitor and provide predictive analytics for the need to increase/decrease its power parameters (network resources, computing resources of the combined cloud infrastructure). As a use case Figure 2 describes the microservices live migration based on the fog resources allocation. The color of the microservices means the different types of services.

At the moment, given the global growth of various applications and their demand from users, new software development architectures have been being changed and proposed. The need for new architectural approaches is dictated by high load, complexity, multicomponent solutions, as well as requirements for the speed of service and development of systems. One of the most actively spreading architectural approaches to the development of high-load applications is microservice architecture. This architecture is designed to solve the problem of creating complex systems by decomposition into independent modules. The level of decomposition (servers on the MVC-pattern model, server modules, classes, objects, databases, or even individual function-methods) is determined in each particular case, project by the development team. However, each of the above-defined modules can be represented in the form of generally accepted interpreted objects: container, virtual machine. For example, if the decomposition is not so low-level, then the architecture of the software that is developed according to the microservice architecture can be represented as the following diagram (Figure 3.)

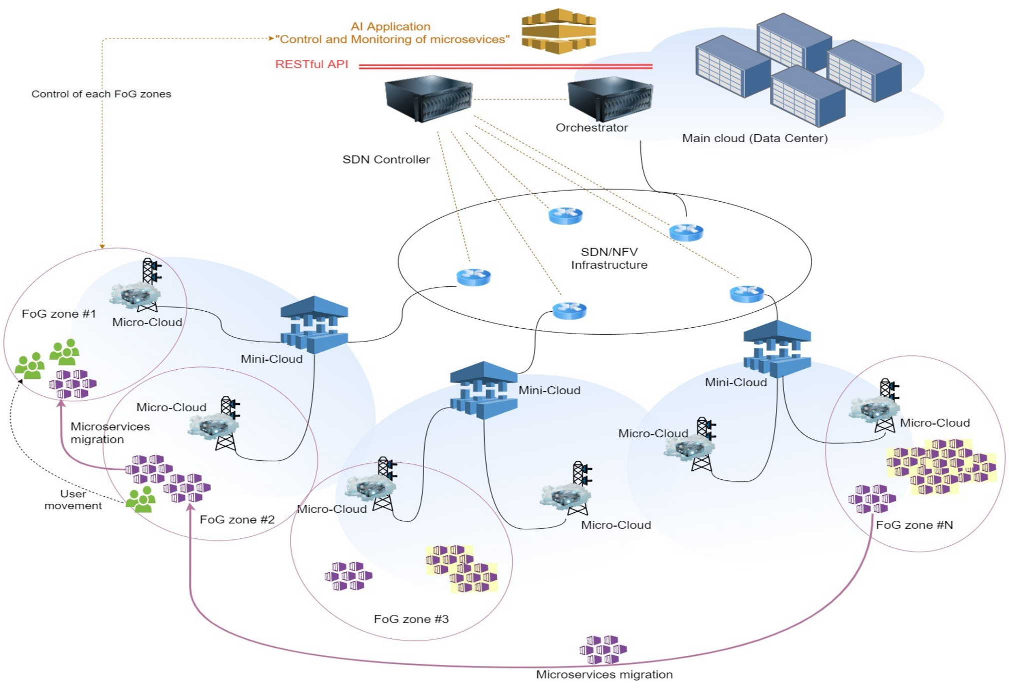

Figure 2. Use case of the framework.

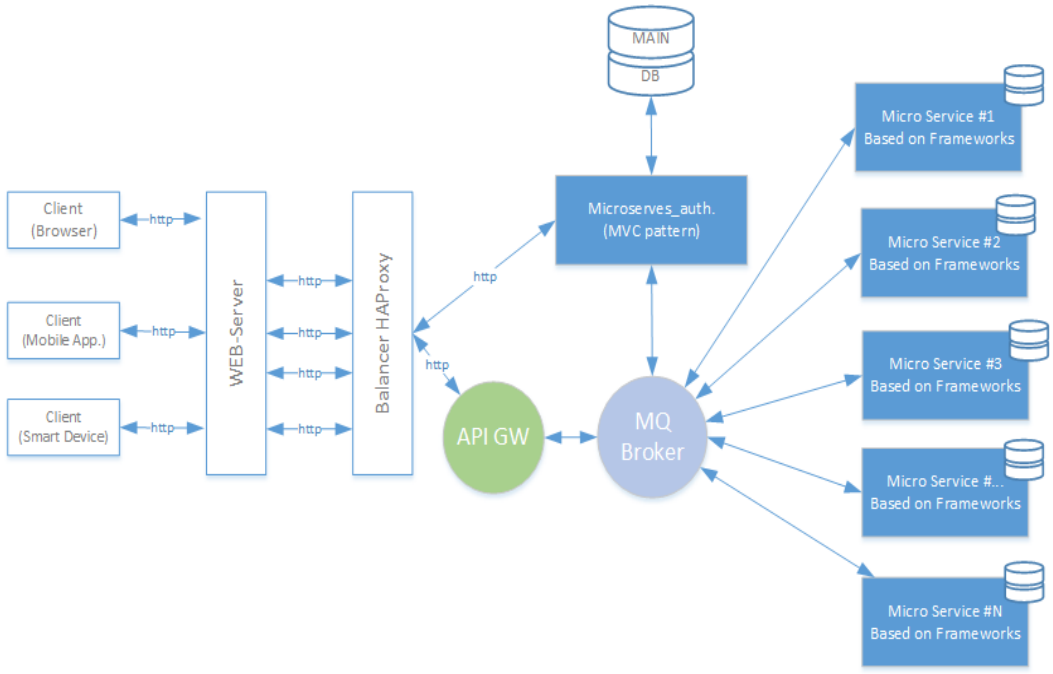
MAIN
DB
Micro Service #1 Based on Frameworks
Micro Service #2 Based on Frameworks
Micro Service #3 Based on Frameworks
Micro Service #... Based on Frameworks
Micro Service #N Based on Frameworks
Microserves_auth. (MVC pattern)
Client (Browser)
Client (Mobile App.)
Client (Smart Device)
http
WEB-Server
Balancer HAProxy
API GW
MQ Broker

Figure 3. Typical microservice high-level functional architecture of the software.

Thus, an application that implements one or another user service can be distributed in the network by modules. Considering the technologies of containerization and virtualization (Linux, VMware, etc.), it is possible to distribute applications, taking into account the needs of each of the decomposed modules, on FoG structures controlled by a multi-tier cloud infrastructure MEC.

Figure 3 summarizes the key elements of software implemented according to a microservice architecture approach. Below are brief descriptions of some of the items.

1. Balancer HAProxy: It is a software module (proxy server) that accepts user requests, performs TCP / HTTP load balancing and distribution of requests between elements

on the Backend. These user requests are generated by various client devices (browser, mobile application, smart device, etc.).

2. API GW-(Application Programming Interface): This is a software module (can be rendered as a separate microservice), which acts as a single accurate reception of requests from external users of the microservices API. API GW receives the request and, based on the URL, determines which microservice should receive/process it. After that, it sends the message through MQ Broker to the recipient microservice, waits for a response and returns it back. The second important function of the GW API is to create and maintain a user session.
3. MAIN DB-(Main Application Database): This database stores information about users and related information, as well as other application metadata.
4. MQ Broker-(Message Broker): This module is required to implement asynchronous communication between system components. Implements the functions of the microservices discovery service in the system. One of the main functions of the module is the implementation of queues: for receiving requests and sending responses to received requests. Each of the microservices connects to a given broker and corresponding message queues. The broker also performs the functions of routing messages based on the metadata transmitted in the message headers (including data about the current user session-authorization token). Each instance of a microservice of a particular type subscribes to listen on the request queue and fetches messages. When sending requests, the recipient is not a specific instance of the microservice, but the type to which it belongs. Since within the framework of the microservice approach, the implementation of each of the microservices can be made in such a way that to increase the performance of the microservice, only copying it is required. The data broker also takes over the load balancing function. In this case, balancing is done implicitly by the microservices themselves, listening to the request queue.

3.2. Entities and Interconnection Model in Framework

Figure 1 in this article showed a generalized architecture of the proposed framework. Within the framework of the architecture under consideration, a number of elements are identified that implement certain computational functions. The structure of the MES implies a hierarchy of elements, with the subordination of the underlying computational layers of the clouds. At the same time, in the structure of fog calculations, as such, there is no strict hierarchy. At the same time, if you correctly organize fog structures, taking into account the technical characteristics and computing capabilities of each of the fog-nodes, their dynamic distribution in space, as well as device-to-device communication (D2D) technologies, it is possible to achieve a greater synergistic effect. In this article, in order to highlight the "communication gateway" with the Internet, it is proposed to use the lower-level server of the fog structure—MicroCloud.

Thus, taking into account the above, it is proposed to highlight the following functional elements of the framework (Figure 4), taking into account the network component of the infrastructure.

The main functional elements are highlighted in blue. Figure 4 also shows possible interactions. For example, device-to-device interaction between fog-nodes, fog-node and micro cloud interaction serving as a fog zone server.

It is worth recalling that within this structure, a dynamic allocation of fog-nodes occurs, as a result of which the resource intensity of each of the fog zones changes. At the same time, on top of this dynamic computing infrastructure, the microservices of the application of a particular service migrate.

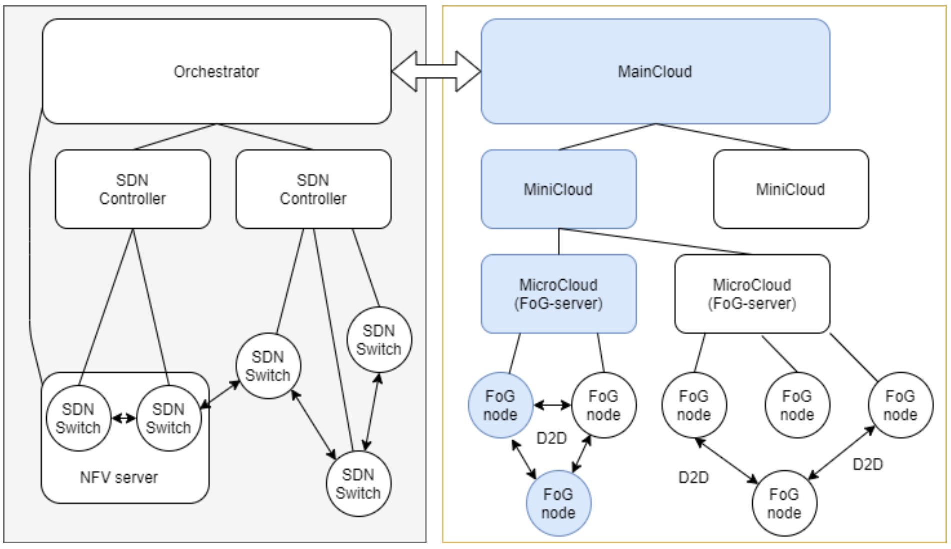

Figure 4. Functional diagram of the elements.

Within the framework of this framework, the following logic of interaction of functional elements is implemented when a new fog-node is connected. The interaction logic is displayed in the form of message diagrams (Figure 5).

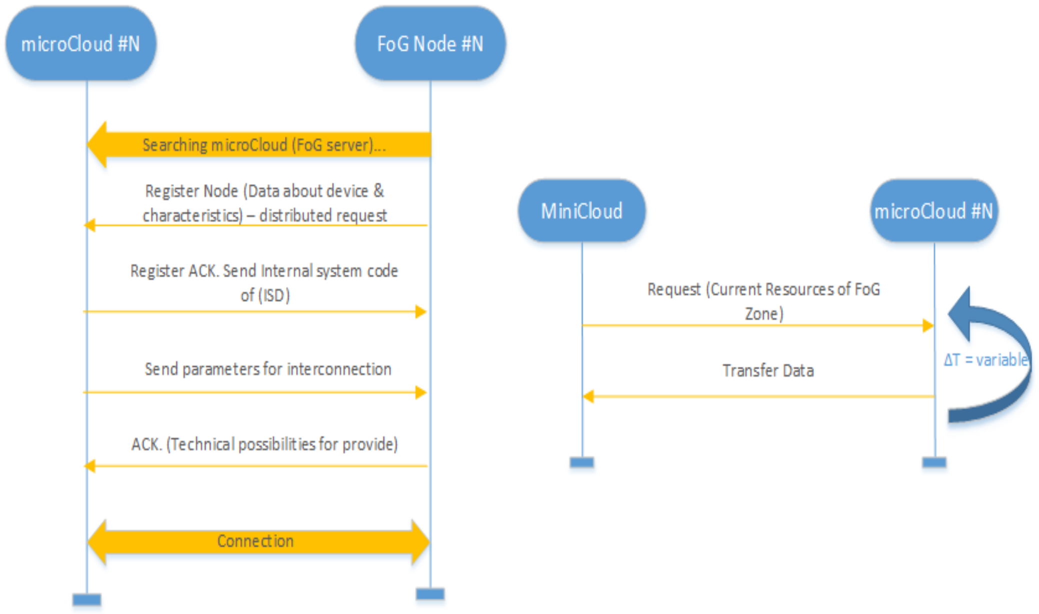
microCloud #N
FoG Node #N
Searching microCloud (FoG server)...
Register Node (Data about device & characteristics) – distributed request
Register ACK. Send Internal system code of (ISD)
Send parameters for interconnection
ACK. (Technical possibilities for provide)
Connection
MiniCloud
microCloud #N
Request (Current Resources of FoG Zone)
Transfer Data
ΔT = variable

Figure 5. Diagrams of interaction of functional elements.

As already indicated above, within the framework of this Framework, dynamic redistribution of fog-nodes is assumed, for this, when switching from one fog zone to another, or during the initial connection, each fog-node sends a broadcast request in order to find the nearest MicroCloud. After detection, fog-node sends data for registration in the detected fog-zone. For registration, fog-node transmits the following data about its computing virtual resources allocated for leasing them as part of the fog zone:

- CPU (cores, frequency of each cores), which presented the device performance and calculated in [Flops].
- RAM.
- ROM.

- Code of active wireless technology (for example: 802.11ac, 5G etc); This code is used for defining the theoretical throughput of the device channel. In this case, the database has a table with relations between wireless codes and theoretical channel throughput.
- Allowed speed for external microservices.
- System of virtualization.
- Supported format of microservices.
- Battery time: battery discharge time to critical value.

In addition, fog-node transmits personal data about the owner to fog-server in order to be identified by the operator. The transfer of information about the owner (structure, obligation, etc.) is not specified in this article and is not a key issue.

After that, the main server of the fog-zone determines the possibility of connecting this fog-node with the specified parameters. In case of a positive decision, fog-server (MicroCloud) sends a confirmation message to the pending fog-node connection. In this message, the fog-server also transmits a unique generated ISC (Internal System Code). This code is required for the subsequent processes of interaction of fog-node with the static cloud MEC infrastructure. Including for the application microservices migration management system, in order to uniquely identify each of the fog-nodes in fog-zones, in conditions of physical movement.

Table 2 provides a comparison on the pros and cons of systems from related works with our proposed system.

Table 2. Comparison systems from related works with our proposed system.

Existing Works	Pros	Cons
[13]	The proposed additional layer for selecting the processing level allows you to move the options for choosing the processing level closer to end users.	The additional level only performs functions for the choice of processing level, but the participation itself does not take part in the processing process.
[14]	The algorithm for choosing mobile devices based on their resources was proposed and described to form a cloudlet cluster.	The article does not consider the possibility of using the fog server for processing user requests. In addition, the article does not consider the problem when it is impossible to form a cluster to handle the user request.
[15]	The article proposes a network hierarchy of the Framework for "foggy" systems, which consists of four levels. The atomization layer is responsible for forming a set of resources for cloud atomization from physical resources.	The proposed architecture uses only the concept of fog computing. Only mobile devices are used to process user requests, and the remote server and MEC servers are not used.
(Proposed System)	Our proposed architecture allows us to process user data at all 4 levels. The formation of a cluster is carried out on the basis of a larger number of parameters.	The proposed network architecture is more complex because it is based on the simultaneous use of MEC and fog technologies (in related works, only one technology was used). The load on the mini-cloud will be greater, since it will be responsible not only for the task of choosing the processing level, but also for processing user requests.

After this message, fog-server transmits connection parameters. As a result, the pluggable fog-node sends a confirmation message to the fog-server about readiness. After that, fog-server receives the rights to use computing and network resources provided by fog-node for its own purposes (aggregation of computing resources).

It is worth noting that further interaction takes place at a higher level of the computational structure, namely: periodic transmission of MicroCloud data about the controlled fog-zone upon request from MiniCloud. At the same time, the frequency of queries is variable which can be determined by the use of one or another algorithm for processing data on fog-zones in order to monitor and manage computing resources.

The structure of the transmitted data fog-server (MicroCloud) is presented in the form of a resource tree in Figure 6.

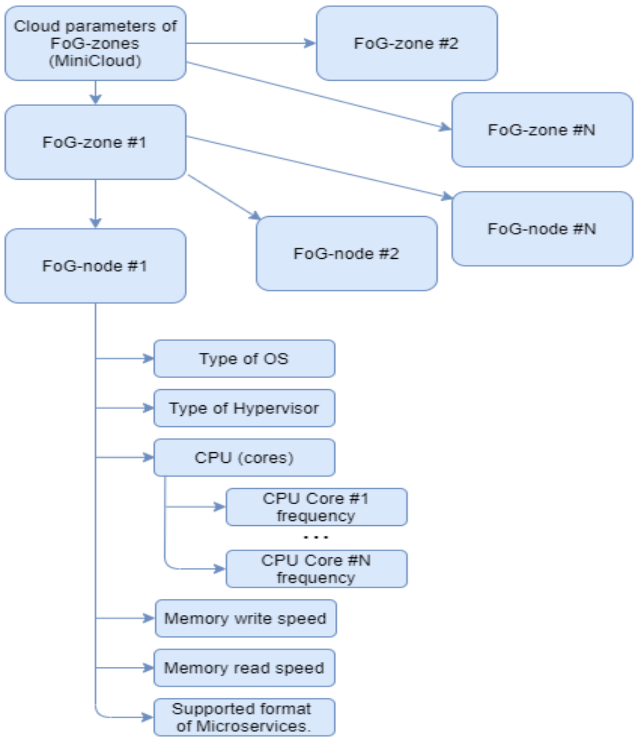

Figure 6. The structure of the transmitted data.

Thus, each MiniCloud has complete information about all controlled fog-zones. This solution allows the development and implementation of a number of monitoring and control algorithms in order to ensure the efficiency of the computing infrastructure, within the framework of the dynamism of fog-nodes, as well as the migration of microservices of the services provided.

4. Proposed Algorithms of Monitoring and Control

In this work, there is a complex, systemic task to determine the center of the accumulation of users (requests from users) of a service, as well as to simultaneously determine the computational potential of a certain foggy computing environment. To solve a number of subtasks within the framework of the above-described problem, it is proposed to use a set of efficient data processing algorithms. The general algorithm of the framework is shown in Figure 7.

This article will look at the parts of the algorithm that are highlighted in orange. Within the general process, these parts are equally complex sub-processes.

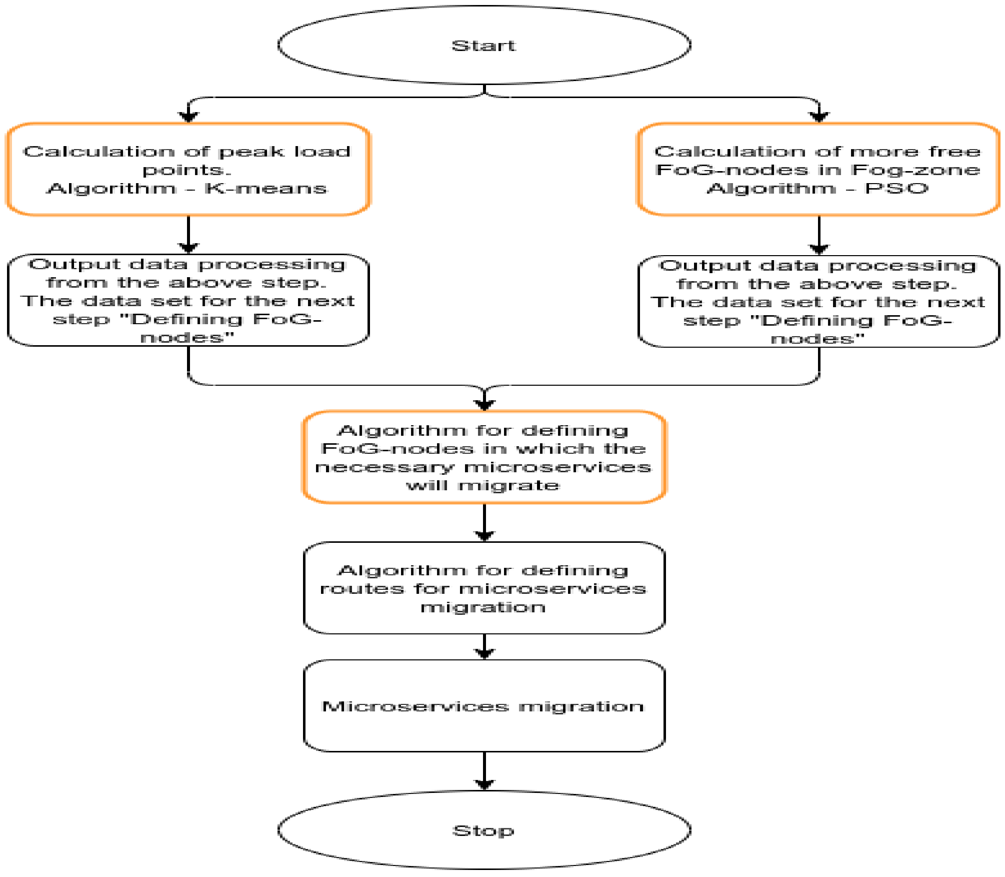
Start
Calculation of peak load points. Algorithm - K-means
Calculation of more free FoG-nodes in Fog-zone Algorithm - PSO
Output data processing from the above step. The data set for the next step "Defining FoG-nodes"
Output data processing from the above step. The data set for the next step "Defining FoG-nodes"
Algorithm for defining FoG-nodes in which the necessary microservices will migrate
Algorithm for defining routes for microservices migration
Microservices migration
Stop

Figure 7. Framework algorithm.

4.1. Determination of the Central Load Point of Users (K-Means-Based Algorithm)

One of the sub-processes is the periodic determination of the epicenter of users of a service. To search for the epicenter, it is proposed to use the k-means clustering algorithms. This algorithm is characterized by simplicity of implementation and speed of execution. The a priori data is the number of clusters (set in advance before the start of the clustering process, and its value has a high impact on the final result). Regarding the specifics of the application of the algorithm in the current task, the number of specified clusters is affected by the metrics collected by the mobile network operator (the number of connections, the average number of connections to the base station per day, etc.) Determining the exact formula, which is describing the dependence of these metrics with the parameter of the number of clusters is not a priority task in this research work. The cluster radii can differ from each other, even if the clusters of service consumers were formed in the same fog-zone, as a result of the execution of the same process with a given number of clusters.

Within one fog-zone, there will be areas with different density of congestion of service consumers [20,21]. Moreover, the sizes of such clusters are also not the same. If we consider the FORELL formal element algorithm as a clustering algorithm, with a predetermined cluster size, then there is a probability that the formed cluster will not completely cover such an area (or the coverage area will be significantly larger than the area of congestion of service consumers). Therefore, the k-means clustering algorithm is used to find user centers. One of the initial conditions within the framework of this article is also the static nature of IoT devices, smartphones and other devices that form fog regions. The radius of the cluster will be the distance between the center and the farthest point in the cluster. In this work, coordinates x, y, z are measured in meters. Three positioning planes were selected in order to take into account the height of the user device in order to correctly form the corresponding clusters. In real conditions, the coordinates of user devices and fog-nodes are proposed to be determined using GPS. The GPS receiver determines the coordinates (latitude and longitude) of the location of three satellites relative to each other and the distance between them and the receiver (determined through the time of transmission of the radio original from the satellites to the receiver).

The algorithm is executed in five stages:

1. Input data are set: x, y and z coordinates of points. The number of clusters and their initial centers are also set.
2. Assigning users (user devices) to the nearest cluster centers. For this, the Euclidean metric of the distance between points in space is used:

$$\sqrt{(x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2} \tag{1}$$

   where x, y and z are the coordinates of the point itself. Determination of centers of mass (the center of mass is determined separately for coordinates x, y, z):

$$C_{mx} = \frac{\sum_{i=0}^{l} x_i}{l}, C_{my} = \frac{\sum_{i=0}^{l} y_i}{l}, C_{mz} = \frac{\sum_{i=0}^{l} z_i}{l} \tag{2}$$

   where l denotes the number of fog-nodes included in the cluster.
3. Comparison of centers of mass and assumed centers of clusters.
4. If the centers of mass and the assumed centers of the clusters are equal, then the centers of the clusters are considered finally determined and all users assigned to them are marked as members of this cluster. If they are not equal, then steps 2–4 are repeated anew, but with the assumed centers of the clusters equal to the centers of mass determined at this iteration.

It should be noted that in this article (at the current stage of the project), when implementing the mathematical model of the algorithm, the mobility of users of services and fog-nodes was not taken into account.

4.2. Service Time Providing through Device Selection (SWARM-Based Algorithm)

One of the sub-processes is the process of determining those fog-nodes that have free computing capabilities and satisfy the conditions for migrating microservices to them with subsequent deployment. Within the framework of this sub-process, it is necessary to solve the optimization problem: for a given function (describing the state of the fog-node through a set of parameters), among all possible values of this function, find a value at which this function takes max values (min values are discarded).

There are many algorithms that are guaranteed to find the extremum of a function, which is a local minimum or maximum near a given starting point. x_0. Such algorithms include, for example, gradient descent algorithms. However, in this problem, it is necessary to find the global maximum of a function that, in a given range of parameters, in addition to one global extremum, has many local extrema. Gradient algorithms cannot cope with the optimization of such a function, because their solution converges to the nearest extremum near the starting point. For issues of finding the global maximum or minimum, so-called global optimization algorithms are used. Swarm intelligence algorithms are one of the classes of these algorithms. In this work, it is proposed to use PSO (practicle swarm optimization). The original PSO algorithm was inspired by the social behavior of biological organisms, specifically the ability of groups of some species of animals to work as a whole in locating desirable positions in a given area, e.g., birds flocking to a food source. This seeking behavior was associated with that of an optimization search for solutions to non-linear equations in a real-valued search space.

Thus, this algorithm is described as follows:

An individual particle i is composed of three vectors: its position in the D-dimensional search space $\underline{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})$, the best position that it has individually found $\underline{p_i} = (p_{i1}, p_{i2}, \ldots, p_{iD})$, and its velocity $\underline{v_i} = (x_{i1}, v_{i2}, \ldots, v_{iD})$. Particles were originally initialized in a uniform random manner throughout the search space; velocity is also randomly initialized.

These particles then move throughout the search space by a fairly simple set of update equations. The algorithm update the entire swarm at each time step by updating the velocity (Equation (3)) and position (Equation (4)) of each particle in every dimension by the following rules [22]:

$$v_{id} = v_{id} + c\epsilon_1(p_{id} - x_{id}) + c\epsilon_2(p_{gd} - x_{id}) \tag{3}$$

$$x_{id} = x_{id} + v_{id} \tag{4}$$

where in the original equations c is a acceleration constant, ϵ_1 and ϵ_2 are independent random numbers in the follow limits [0, 1], $\underline{p_{i_d}}$ is the best position passed by all particles, $\underline{p_{g_d}}$ is the position found by any neighbor of the particle. The update process is summarized in Algorithm 1.

Algorithm 1 PSO Update Process

```
1: for each time step $t$ do
2:   for each particle $i$ in the swarm do
3:     Update position $\underline{x_t}$ using Equations (3) and (4).
4:     Calculate particle fitness $f(\underline{x_t})$
5:     Update $\underline{p_i}$, $\underline{p_g}$.
6:   end for
7: end for
```

Particle velocities in this original algorithm were clamped at a maximum value $vmax$. Without the clamping in place the system was prone to entering a state of explosion, wherein the random weighting of the ϵ_1 and ϵ_2 values caused velocities and thus particle positions to increase rapidly, approaching infinity. The v_max parameter prevented the system from entering this state by limiting the velocity of all particles to that value.

Thus, it is necessary to define the parameters describing each of the investigated fog-nodes. At the same time, some of the parameters are evaluated at the level of comparisons with limit values, for example, the amount of allocated logical RAM required for the microservice to work. For optimization, parameters were defined that describe fog-node in terms of ensuring quality of service. Globally, the task is as follows: it is necessary to maintain the service time by choosing of the fog-node to which the microservice needs to be migrated. Thus, the minimized fitness function looks like this:

$$T = \sum_{i=0}^{n} w_i TimeSlot_i \tag{5}$$

Where T is a total parameter, W_i is the weight of the 1-st parameter, $TimeSlot_i$ is the parameter value, n is the total number of parameters. In this article will used the following two parameters: $TimeSlot_1$ (network delay), $TimeSlot_2$ (microservice request processing time). For each of the parameters, weight (W) is 0.5.

Time parameters are measured in milliseconds (ms). In general, the fitness function is:

$$T = 0.5TimeSlot_1 + 0.5TimeSlot_2 \tag{6}$$

The first parameter is determined via a time tracker in the framework. The second parameter is sent by the device—the processing time of the task.

5. Framework Modeling

Regarding the considered framework algorithm (Figure 7) in the previous paragraph, this paper considers two algorithms: determining the centers of user congestion (user devices) and searching for more free fog-nodes to which the necessary microservices will be migrated. For a practical study of the proposed framework and the two algorithms under consideration, the following use case was developed, based on the city model. This model includes: two high-rise office buildings, a fast food cafe on the first floors of the buildings, an intersection.

5.1. K-Means-Based Algorithm Results

To test the proposed algorithm, the above algorithm was implemented in python. The simulation settings employed for K-means-based Algorithm are summarized in Table 3. Within the framework of the implemented software, at the first step, data on user devices was generated.

Table 3. Simulation parameters.

Cluster	**1**	**2**	**3**	**4**	**5**
Number of Users	10	50	15	30	50
Congestion of users	in a fast food cafe on the ground floors	on higher floors 415 of an office building	on higher floors of an office building	in a fast food cafe on the ground floors	on higher floors of an office building

The following input data was used in the algorithm:

1. The total number of clusters is 5.
2. Presumptive centers were randomly assigned according to the principle of places in a space with a large number of possible users.

Thus, using the "random()" function, the coordinates of the devices of the regions under study (clusters) were generated. Where, in cluster 1, 10 user coordinates are generated, in the 2nd cluster—50, 3rd—15, 4th—30, 5th—50.

For clarity of the proposed model, fog-nodes were also generated. The resulting clusters are shown in Figure 8, where the corresponding clusters and their numbers are highlighted.

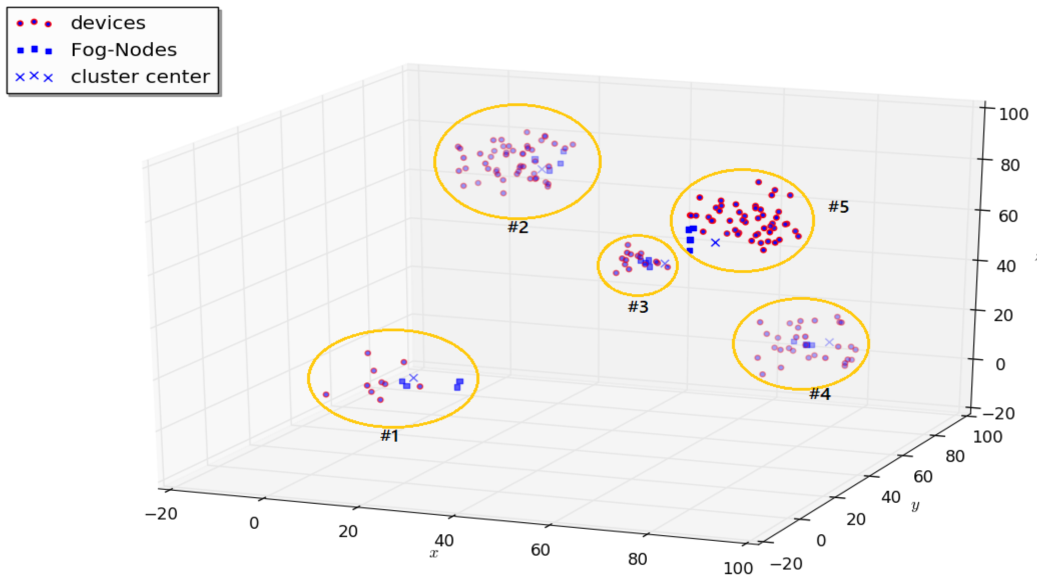

Figure 8. Generated clusters.

From the point of view of the practical model, clusters 1 and 4 reflect the congestion of users in a fast food cafe on the ground floors, as well as users on the next floors of an office building. Clusters 2, 3 and 5 also display the congestion of users on higher floors of an office building, where, for example, there is an openspace, which in practice means a higher density of people per square meter.

At the modeling stage, according to the k-means algorithm, the initial user centers were also determined (possible centers—in fact, a stochastically selected point was indicated in a possible area of user congestion).

At the next stage of modeling, the k-means algorithm came into operation in order to determine the center of mass (center of users), the radius of their scatter relative to their location. The result of the algorithm is shown in Figures 9 and 10.

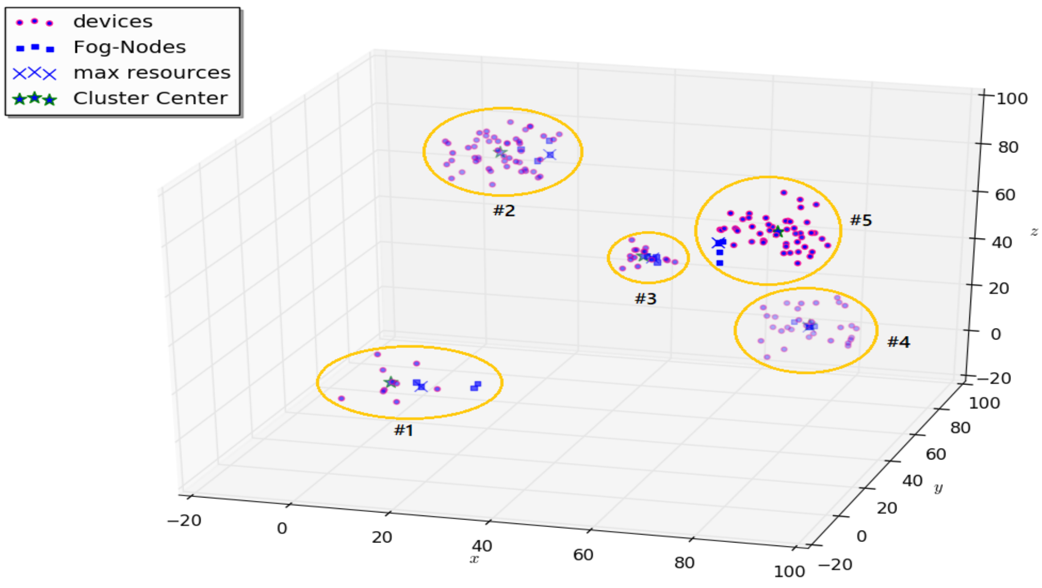

Figure 9. Algorithm results (position 1).

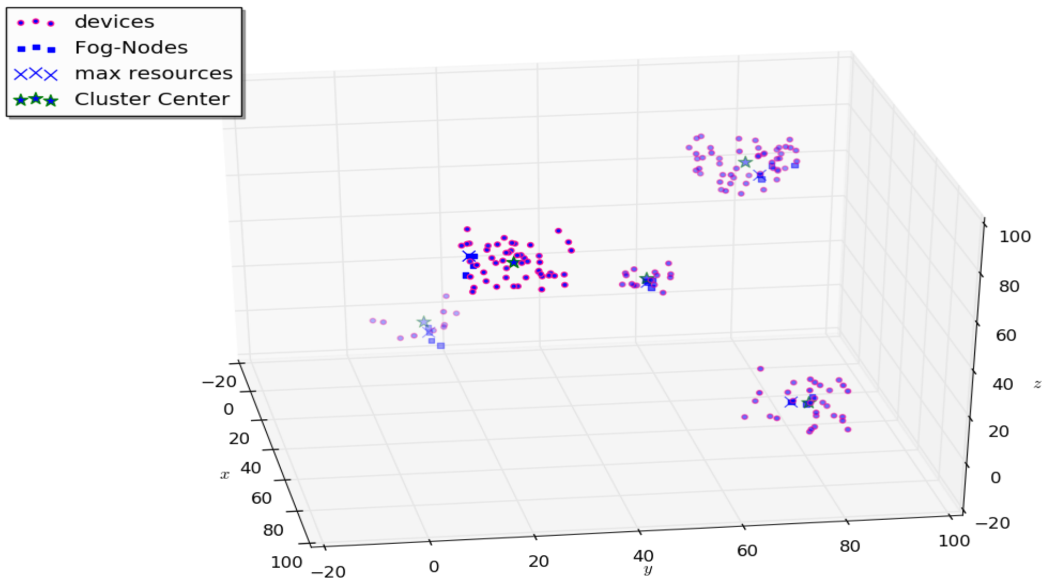

Figure 10. Algorithm results (position 2).

Thus, as a result of modeling, the centers of the clusters were determined (Figures 9 and 10), as well as the radii. The cluster radii are shown in the bar graph (Figure 11).

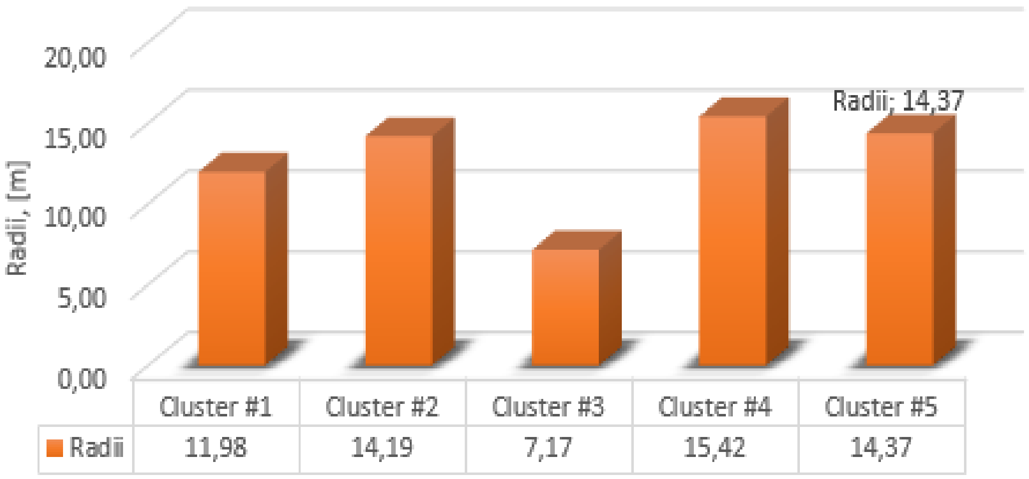

Figure 11. Radii of clusters.

5.2. *Swarm-Based Algorithm Results*

To test the proposed algorithm, the above SWARM algorithm was developed on python. Within the framework of the implemented software, at the first step, data on user devices was generated.

According to the defined fitness-function and described PSO algorithms, results were received for the 5th cluster , which are presented in the following Figures 12 and 13. In the 5th cluster, 50 devices with their own timing characteristics, which were generated within the following limits: $TimeSlot_1 \in [0.5, 10]$ ms, $TimeSlot_2 \in [0.2, 2]$ ms.

After processing by the PSO algorithm, the device with the minimum time characteristic was found. It is 13th device with $TimeSlot_1 = 1.18$ ms, $TimeSlot_2 = 0.76$ ms and Fitness function = 0.97. In addition, RTT for this device is = $TimeSlot_1 + TimeSlot_2$ = 3.12 ms. The 13rd device presented on the following picture as cross.

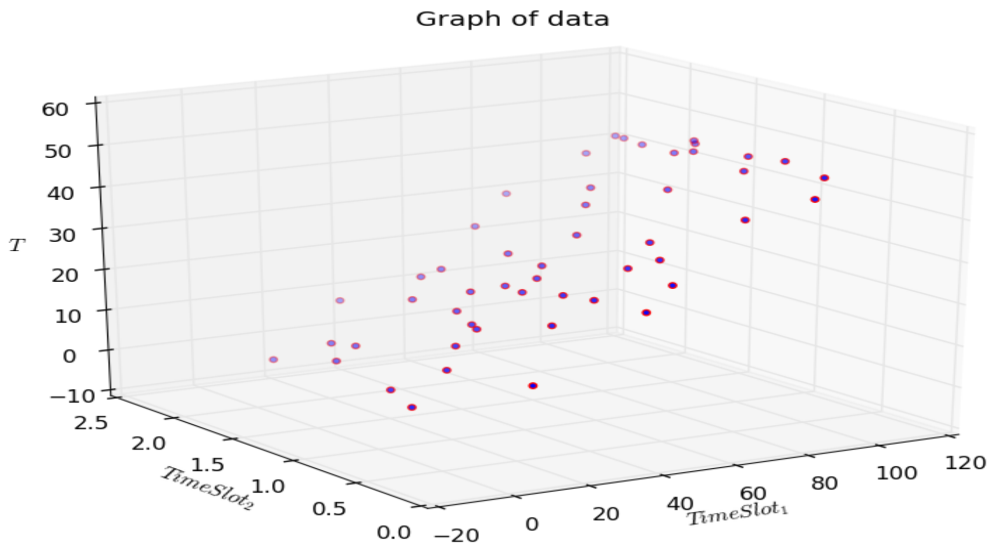

Figure 12. Fitness function value for each device.

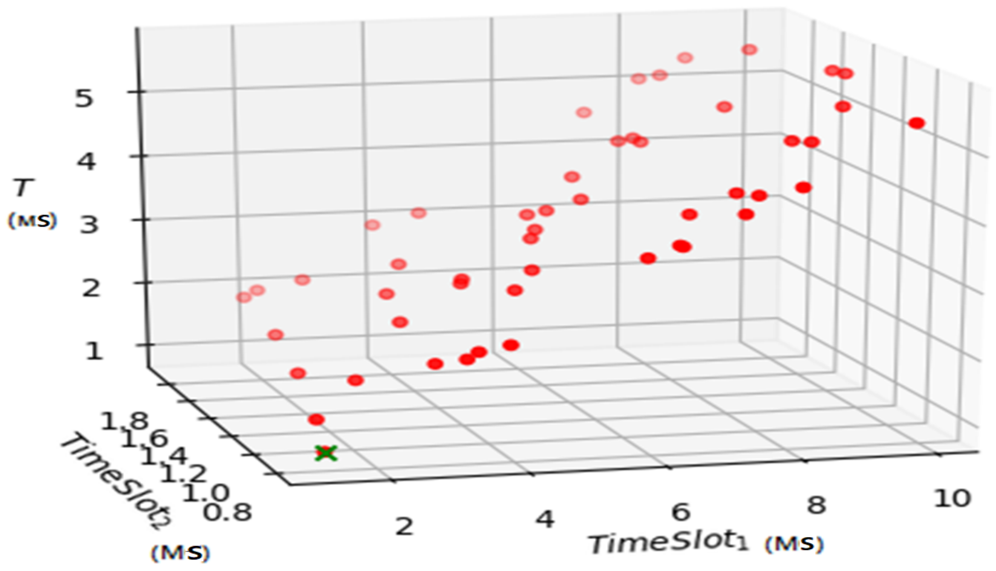

Figure 13. Result of PSO modeling.

In addition, based on the fitness-function we can separate the fog- devices on the micro-clusters for the services with the necessary quality. An example of these micro-clusters separating is displayed on Figure 14, where each of micro-clusters provides the necessary quality characteristics for services.

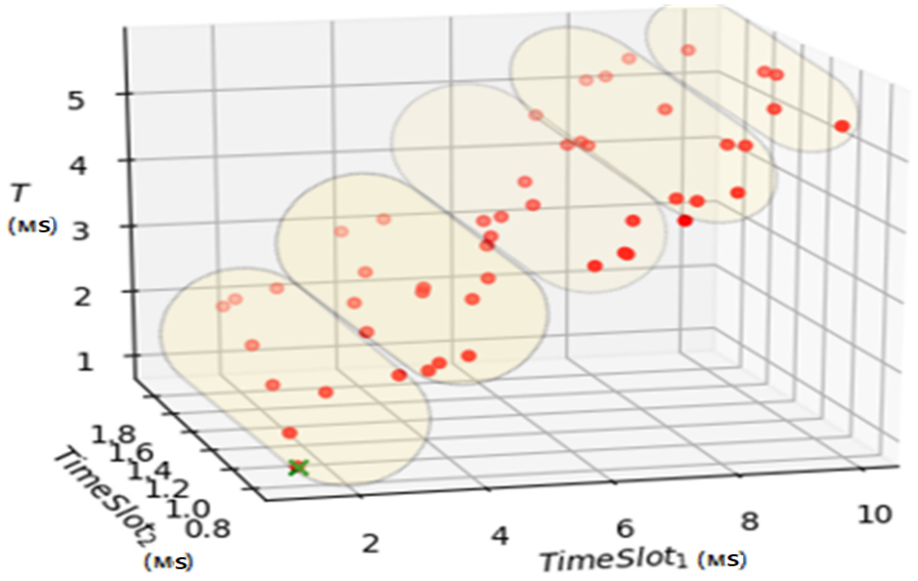

Figure 14. Micro-clusters separating.

6. Conclusions

A novel distributed and dynamic fog (DD-fog) framework has been proposed in this paper for deploying services based on MEC and fog infrastructure in accordance with user dynamics. In addition, migrating the microservices onto the necessary fog-node is considered as one of the key feature of this framework which can provide with a stable quality level for each service. Moreover, two K-means and SWARM-based algorithms were investigated for determining the centers of high demand and finding the fog-node that is more consistent with the service requirements. Furthermore, the fog separation on microclusters was suggested which is based on the fitness-function metrics. Finally, simulation results proved that the proposed framework could reduce the execution time of the microservice function due to the rational allocation of resources by up to 70%.

For our future work, we plan to build a new predictive model of user moving and service dynamics, in which the results of this paper are considered. On the next step of this project, expected to build the predictive model of user routes and service dynamics, taking into account presented results. For example, the following tasks monitoring of the consumer demand (AI), taking into account their routes monitoring of the load on the services and their components (microservices) prediction of user routes and their demand (AI) based on the monitoring prediction of routes for microservices migration (via edge network and D2D) (AI) based on the biological algorithms.

Author Contributions: Conceptualization, V.A. and K.V.; methodology, V.A.; software, A.M.; validation, I.A.E. and A.K.; formal analysis, K.V.; investigation, A.M.; resources, A.K.; writing—original draft preparation, V.A.; writing—review and editing, I.A.E.; supervision, A.M. and A.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research is based on the Applied Scientific Research under the SPbSUT state assignment 2021 and was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-track Research Funding Program.

Data Availability Statement: The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Park, J.H.; Rathore, S.; Singh, S.K.; Salim, M.M.; Azzaoui, A.E.; Kim, T.W.; Pan, Y.; Park, J.H. A Comprehensive Survey on Core Technologies and Services for 5G Security: Taxonomies, Issues, and Solutions. *Hum.-Centric Comput. Inf. Sci.* **2021**, *11*, 22.
2. Khayyat, M.; Alshahrani, A.; Alharbi, S.; Elgendy, I.; Paramonov, A.; Koucheryavy, A. Multilevel service-provisioning-based autonomous vehicle applications. *Sustainability* **2020**, *12*, 2497. [CrossRef]
3. Alshahrani, A.; Elgendy, I.A.; Muthanna, A.; Alghamdi, A.M.; Alshamrani, A. Efficient multi-player computation offloading for VR edge-cloud computing systems. *Appl. Sci.* **2020**, *10*, 5515. [CrossRef]
4. Guerrero, C.; Lera, I.; Juiz, C. Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. *Future Gener. Comput. Syst.* **2019**, *97*, 131–144. [CrossRef]
5. Duc, T.L.; Leiva, R.G.; Casari, P.; Östberg, P.O. Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–39. [CrossRef]
6. Rec, I. ITU-R M. 2083-0. In *IMT Vision—Framework and Overall Objectives of the Future Development of IMT*; International Telecommunication Union, ITU-R, Genève, 2020.
7. Gao, J.; Li, W.; Zhao, Z.; Han, Y. Provisioning big data applications as services on containerised cloud: a microservices-based approach. *Int. J. Serv. Technol. Manag.* **2020**, *26*, 167–181. [CrossRef]
8. Alaasam, A.B.; Radchenko, G.; Tchernykh, A.; Compeán, J.G. Analytic Study of Containerizing Stateful Stream Processing as Microservice to Support Digital Twins in Fog Computing. *Program. Comput. Softw.* **2020**, *46*, 511–525. [CrossRef]
9. Herrera, J.L.; Galán-Jiménez, J.; Berrocal, J.; Murillo, J.M. Optimizing the Response Time in SDN-Fog Environments for Time-Strict IoT Applications. *IEEE Internet Things J.* **2021**, *8*, 17172–17185. [CrossRef]
10. Al-Ansi, A.; Al-Ansi, A.M.; Muthanna, A.; Elgendy, I.A.; Koucheryavy, A. Survey on Intelligence Edge Computing in 6G: Characteristics, Challenges, Potential Use Cases, and Market Drivers. *Future Internet* **2021**, *13*, 118. [CrossRef]
11. Zhang, W.Z.; Elgendy, I.A.; Hammad, M.; Iliyasu, A.M.; Du, X.; Guizani, M.; Abd El-Latif, A.A. Secure and Optimized Load Balancing for Multitier IoT and Edge-Cloud Computing Systems. *IEEE Internet Things J.* **2020**, *8*, 8119–8132. [CrossRef]
12. Elgendy, I.A.; Zhang, W.Z.; Zeng, Y.; He, H.; Tian, Y.C.; Yang, Y. Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2410–2422. [CrossRef]
13. Tran, T.X.; Hajisami, A.; Pandey, P.; Pompili, D. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. *IEEE Commun. Mag.* **2017**, *55*, 54–61. [CrossRef]
14. Li, Y.; Anh, N.T.; Nooh, A.S.; Ra, K.; Jo, M. Dynamic mobile cloudlet clustering for fog computing. In Proceedings of the 2018 international conference on electronics, information, and communication (iceic), Honolulu, HI, USA, 24–27 January 2018; pp. 1–4.
15. Ningning, S.; Chao, G.; Xingshuo, A.; Qiang, Z. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Commun.* **2016**, *13*, 156–164. [CrossRef]
16. Volkov, A.; Khakimov, A.; Muthanna, A.; Kirichek, R.; Vladyko, A.; Koucheryavy, A. Interaction of the IoT traffic generated by a smart city segment with SDN core network. In Proceedings of the International Conference on Wired/Wireless Internet Communication, Moscow, Russia, 25–29 September 2017; pp. 115–126.
17. Volkov, A.; Muhathanna, A.; Pirmagomedov, R.; Kirichek, R. SDN approach to control internet of thing medical applications traffic. In Proceedings of the International Conference on Distributed Computer and Communication Networks, Moscow, Russia, 25–29 September 2017; pp. 467–476.

18. Volkov, A.; Proshutinskiy, K.; Adam, A.B.; Ateya, A.A.; Muthanna, A.; Koucheryavy, A. SDN load prediction algorithm based on artificial intelligence. In Proceedings of the International Conference on Distributed Computer and Communication Networks, Moscow, Russia, 23–27 September 2019; pp. 27–40.
19. Muthanna, A.; Volkov, A.; Khakimov, A.; Muhizi, S.; Kirichek, R.; Koucheryavy, A. Framework of QoS management for time constraint services with requested network parameters based on SDN/NFV infrastructure. 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Moscow, Russia, 5–9 November 2018; pp. 1–6.
20. Aljubayri, M.; Yang, Z.; Shikh-Bahaei, M. Cross-layer multipath congestion control, routing and scheduling design in ad hoc wireless networks. *IET Commun.* **2021**, *15*, 1096–1108. [CrossRef]
21. Ibrar, M.; Wang, L.; Muntean, G.M.; Shah, N.; Akbar, A.; Qureshi, K.I. SOSW: scalable and optimal nearsighted location selection for fog node deployment and routing in SDN-based wireless networks for IoT systems. *Ann. Telecommun.* **2021**. [CrossRef]
22. Bratton, D.; Kennedy, J. Defining a standard for particle swarm optimization. In Proceedings of the 2007 IEEE swarm intelligence symposium, Honolulu, HI, USA, 1–5 April 2007; pp. 120–127.