Computer Systems Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2023.038545

ARTICLE

A Multi-Objective Genetic Algorithm Based Load Balancing Strategy for Health Monitoring Systems in Fog-Cloud

Hayder Makki Shakir, Jaber Karimpour* and Jafar Razmara

Department of Computer Science, Faculty of Mathematical Sciences, University of Tabriz, Tabriz, 51368, Iran

*Corresponding Author: Jaber Karimpour. Email: Karimpour@tabrizu.ac.ir

Received: 17 December 2022 Accepted: 18 April 2023 Published: 26 January 2024

ABSTRACT

As the volume of data and data-generating equipment in healthcare settings grows, so do issues like latency and inefficient processing inside health monitoring systems. The Internet of Things (IoT) has been used to create a wide variety of health monitoring systems. Most modern health monitoring solutions are based on cloud computing. However, large-scale deployment of latency-sensitive healthcare applications is hampered by the cloud's design, which introduces significant delays during the processing of vast data volumes. By strategically positioning servers close to end users, fog computing mitigates latency issues and dramatically improves scaling on demand, resource accessibility, and security. In this work, we describe a new load-balancing strategy based on the Strength Pareto Evolutionary Algorithm 2 (SPEA2) for distributing work among fog nodes in a large-scale health monitoring system. We ran comprehensive simulations in the iFogSim toolkit to verify the efficacy of the proposed method, comparing the results to the cloud-only implementation, the Fog Node Placement Algorithm (FNPA), the Load Balancing (LAB) scheme, and the Load Balancing Scheme (LBS) in terms of latency and network utilization. The proposed health monitoring system deployment drastically decreases latency and network use in comparison to cloud-only, FNPA, LAB, and LBS schemes.

KEYWORDS

Fog computing; fog node placement algorithm; load balancing; load balancing scheme; Strength Pareto Evolutionary Algorithm 2

1 Introduction

The Internet of Things (IoT) has gradually integrated itself into our everyday lives. The IoT has demonstrated its importance and potential in a variety of fields, including smart cities, smart home systems, and healthcare systems, among others [1]. As with other application areas, the IoT has significantly revolutionized the healthcare sector. The primary objective of healthcare applications is to continuously monitor a patient's health. Therefore, time-sensitive and real-time treatments play an important role in healthcare. Numerous healthcare architectures have been proposed, the majority of which include IoT and cloud computing. The evolution of cloud computing has made it a viable option for data storage applications and data processing [2].

However, cloud computing has significant obstacles, such as traffic congestion, data transmission delays, process of huge volumes of data, etc. Most of these problems originate from the fact that

cloud servers are often located in inconveniently distant locations from IoT gadgets [3]. The critical nature of the industry makes it evident that delays are not an option for healthcare apps. So, it is impractical to apply ordinary cloud computing services to gather and make analysis of patient medical data across a vast geographic area [4] due tothe significant communication delays and high network consumption involved. Consequently, fog computing has evolved as a new paradigm to address the aforementioned fundamental difficulties of traditional cloud computing [5]. Fog nodes are the distributed computer equipment that makes up a fog network. In fog computing geographically distributed architecture, heterogeneous devices are connected to the network for the provision of computing and storage resources. Fog computing is an architecture that provides a more secure and adaptive method for the management of data with low bandwidth use. Wireless communications will greatly improve performance and lower healthcare expenses because of technological advancements in health monitoring systems and the Internet of Things. The Internet of Things makes it easier to monitor these individuals by offering low-cost home monitoring systems that can identify early signs of health decline and enable quicker response and treatment.

A sizable amount of access to network resources is required in a cloud-based health monitoring system because user IoT devices are expected to transfer a considerable amount of sensor-generated data to the cloud's server [6]. Delay is an essential feature for time-critical applications like health monitoring systems. The more the system is used, the more important it becomes. The paradigm of fog computing is crucial in addressing the aforementioned issues. By placing components at the network's periphery, or "fog", latency is drastically cut down. Since fog computing is just a more advanced type of cloud computing, the cloud layer must exist alongside it and give support as needed [7]. Together with local processing and storage, fog computing may also manage a distributed system of devices and sensors [8]. Thus, it appears that IoT systems with specific requirements might benefit more from fog computing. The latency of the application can be decreased by including fog-based computing in system design, which reduces the amount of data transmission to the cloud server.

For instance, many scholars have claimed that using a fog computing architecture as opposed to a cloud computing design reduces latency [9]. Fog computing is perfect for IoT applications because it lowers network traffic and enhances scalability. The use of fog-enabled devices [7] significantly lowers network use, which is nevertheless an essential factor in real-time applications. With an increase in the number of connected devices, fog servers are increasingly taxed by the transmission of real-time data used in advanced applications. It is crucial to balance the demand for fog nodes to effectively support applications. To increase the performance and reliability of the applications, load balancing is the process of distributing application traffic among several servers. If one server gets overloaded by an influx of client requests, some of the work can be redistributed to the next available server. Utilizing this method of workload distribution allows for the optimal use of energy and resources [5].

In this research paper following key contributions:

- We outline an architecture that is fog-based for the ecosystem that is for monitoring health. The suggested fog-based health monitoring architecture determines a person's health condition by analysis of data sent and received via sensors.
- There are three levels in the proposed design. Monitoring vital signs including core temperature, heart rate, and pulse are all handled by sensors in the outermost layer. All IoT gadgets are linked to fog nodes in the fog layer, an intermediate layer. First-tier IoT devices transmit sensor data to fog nodes, which in turn determine the severity of the patient's condition and report their findings to a cloud server in the third layer, where they are stored indefinitely. The fog nodes also relay patient's health status to their mobile devices.

- The suggested design of the system monitoring for health aims to give patients that have continuous access to real-time medical support, and fog computing is beneficial for implementing time-sensitive applications. With fog computing, resources are located close to end users, making it possible to manage the vast amounts of data produced by end-user devices.
- Since fog computing necessitates real-time, efficient data processing, the proposed system is a natural fit. The processing and storage capabilities of fog servers are inferior to those of cloud servers.

Increased data flow in huge systems places additional stress on the fog server [10] because of an increase in the number of user requests. In the projected widespread rollout of a fog-based health monitoring system, the increasing number of patient requests for a given fog node will place an increasing strain on that fog node. As a result, the response time and latency will increase since that fog node will get overloaded while the others will likely remain dormant. Because of the time-sensitive necessity of systems for monitoring health, we suggested a Load Balancing Scheme based on the Strength of Pareto Evolutionary Algorithm 2 (SPEA2). This scheme distributes the load to neighboring fog nodes in a way that decreases latency and network utilization. We assume that, as of [11], there may be traffic and compute lag in IoT data flows. In the suggested method, the IoT device chooses a fit fog node to decrease latency in the health monitoring system. Through simulations with the iFogSim toolbox, the efficacy of the proposed technique was evaluated in comparison to other benchmark strategies, including cloud-only implementation, Load Balancing Scheme (LB Scheme) (LBS), Load Balancing Scheme (LAB Scheme), and Fog Node Placement Algorithm (FNPA). The experimental outcomes demonstrated the technique's efficient network use and delay management.

We organized the papers follows: In Section 2, recent studies on the design of load balancing in fog-based systems and systems for monitoring health are discussed. While Section 3 details our load-balancing method, and Section 4 discusses the suggested design for health monitoring systems. Section 5 presents the results and discussions of the experiment, and Section 6 describes the experimental setting. In Section 7, we used Performance Comparison to discuss the article and our work, and conclusion and future work.

2 Related Work

HealthFog is a smart healthcare system based on fog computing that was presented by Farahani et al. [12] for the detection of heart illness. The suggested method utilizes hardware-based software to facilitate rapid and secure information transfer. Similarly, Kunal et al. [13] proposed a fog-based eHealth application for monitoring patients' health by collecting physiological indicators and environmental data (such as light and air quality). By combining sensor nodes, a cloud server for processing in the event of an abnormal health condition, and fog nodes for parametric health control, Mukherjee et al. [14] presented a three-tier Internet of Health Things (IoHT) architecture with mobility awareness. In [12,14], authors compared cloud-based health monitoring systems to their suggested designs for fog computing-based systems, demonstrating the validity of both approaches. Instead, we compared two earlier implementations of fog-based telemedicine with our proposed fog-based telemedicine and fog-enabled telemedicine systems. The cloud-only implementation option is provided alongside the FNPA and LAB schemes. Taneja et al. [15] proposed a health-tracking application built on top of the fog layer, which processes and transmits sensor-collected physiological data. da Silva et al. [16] also provided an architecture for real-time data processing, which is essential in fields like healthcare.

However, no simulations were run to evaluate the suggested structures by [15] or [16]. For unaccompanied and elderly adults, Fan et al. [17] and Tuli et al. [18] created fog-based health surveillance systems. The J48Graft decision tree was used by Vilela et al. [19] to provide a fog-enabled healthcare system for tracking blood glucose levels and predicting an elevated risk for diabetes. Despite using simulations to test their suggested solutions, the researchers in [17–19] did not provide enough information on latency and network usage performance measurements. Mukherjee et al. [20] provided a fog computing-enabled health monitoring system for diabetics suffering from cardiovascular diseases. The COVID-SAFE framework was created by Ben Hassen et al. [21] to reduce the risk of corona exposure. Fog nodes that are powered by Machine Learning (ML) technologies process and analyze data. A fog computing-based home hospitalization system was presented by Badidi et al. [22]. With this approach, patients can receive care at home, where doctors can keep an eye on their environment's health. The authors of [20–22] did not test their suggested network utilization and delay approaches. The IoT-enabled healthcare system that Saidi et al. [23] suggested manages the requests of patients from various cities through several fog nodes.

If the patient's condition is serious, the request will send directly to a cloud server; otherwise, it is processed by a fog node. If the patient's condition changes, the request is sent to the cloud server from the fog node. However, no real-world cloud- or fog-based healthcare system deployment is used to compare results, only simulations of those systems. Debauche et al. [24] designed an architecture for a fog-enabled health monitoring system in which fog nodes analyze data according to a perfect task-scheduling algorithm. In [25], we find yet another system architecture for managing patient health records that makes use of cloud and fog computing. With the help of the proposed task allocation mechanism, the collected data can be processed more efficiently. The effectiveness of the architecture and algorithm described in [24,25] was measured by comparing simulation results with cloud-only architecture but without any fog-based architecture. A three-layer concept for a remote pain monitoring system was proposed by Nguyen Gia et al. [26] in which a digital signal processing is used by the fog node to identify discomfort. The suggested solution resulted in minimal latency as opposed to cloud-based implementation. However, the suggested approach won't scale well as the patient population increases.

This is because a single fog node handle sells of the hospital's data. For the three levels of the suggested design, a load-balancing mechanism was put forth by Vedaei et al. [27]. The number of activities taken is constrained by a threshold that is assigned to the fog layer. Work requests are sent to the cloud server at the top layer when this threshold is reached. The outcomes of the simulation showed that the suggested approach decreases turnaround time generally. In their paper, Ben Hassen et al. [28] presented a multi-tenant load distribution algorithm for fog environments. (MtLDF). Compared to Delay-Driven Load Distribution, MtLDF has been proven to be more effectively distributed loads (DDLD). This paper presents DRAM, a dynamic resource allocation method, developed by Khattak et al. [29] for load balancing in systems that are fog-based. A related approach to resource allocation is provided that has shown promise in terms of resource usage but does not consider latency in its performance assessment. It involves statically allocating resources and transferring dynamic services. A method for allocating work to fog nodes based on their processing capability and power consumption has also been presented by Paul et al. [30].

Simulations show that the suggested technique significantly reduces application latency, network utilization, and energy usage. For efficient use of available resources, Abdelmoneem et al. [31] developed a FNPA that connects Internet of Things (IoT) gadgets to the fog node that is geographically nearest to them (CPU, RAM, and Bandwidth). When compared to both cloud-only implementation and the fog-node that have minimum distance approach, the FNPA demonstrated much lower latency,

execution cost, and network utilization. Network delays due to the number of service requests and compute delays due to the need to provide resources for service requests are two potential sources of disruption to data flows from IoT devices, as described by Hassan et al. [32]. Workload Allocation (WALL) and Application-aware workload Allocation (AREA) are two approaches proposed for hierarchical cloud computing by the authors of [32,33]. Every one of the described approaches allocates requests from IoT users to suitable cloudlets, which eliminates network and computation delays and reduces the average response time of applications. Mukherjee et al. in their paper [11], proposed a LAB Scheme for systems that are fog computing-based to lessen the lag time of IoT dataflow. The problem is solved by the LAB Scheme, which routes devices to the most suitable BSs based on the tasks to be performed.

Anam and Hassan introduce a Load Balancing Scheme (LBS) that takes fog overloading and user transfers between fog nodes into consideration. In addition, they compare their method to earlier works and demonstrate its effectiveness.

Many of the aforementioned publications [12–14,24,25,30] just compared the simulation results of their preferred fog-based methodology against cloud-based implementation. Other articles [20–22] that analyzed proposed architectures never mentioned performance indicators like latency and network use. Some scholars do not consider all the relevant factors; for example, in [29], the effect of the proposed method on latency is not discussed. While in [11,27,28,32,33] network use is not factored into performance evaluations. Some of the preceding articles [20,21] failed to conduct experiments to test the efficacy of their proposed methods. Unlike previous research [34,35], there was a use of Internet of Things applications with a load balancer in determining sources and also reducing consumption and speed of response, but the largest speed of response to data for the patient in some critical cases was not addressed. We ran extensive simulations with a wide range of configurations of topology and collated the results to those of three more fog-based implementations (FNPA, LAB, and LBS Scheme) and a cloud-only version.

3 System Architecture

In this part, we discuss the three-tier architecture of a fog-based health monitoring system. Sensors that are worn by the patient monitor and communicate vital signals including core temperature, heart rate, and pulse rate to fog nodes as the initial layer of the suggested architecture. Fog nodes make up the architecture's second level. The information gathered from the patient's IoT devices is analyzed by the fog nodes, and the result is relayed to the patient's device. Fog nodes are located near the network's edge to ensure that patients receive a quick reaction in real-time. As the recommended architecture's capstone, the cloud datacenter is where the patient's health status data is kept. Gateways form the connection between the cloud server and the fog nodes. This architecture relies on a cloud server that is primarily designed to provide access to huge data centers. Fig. 1 depicts the fog computing-based architecture of this our proposed system.

The first layer of the suggested health monitoring system is made up of IoT devices. The IoT gadget's sensors record the patient's body temperature, blood pressure, and pulse rate and send this information to a higher-level cloud device for analysis to determine whether or not the patient is in a severe situation. The intermediary layer is between the cloud layer and the Internet of Things layer. Data from sensors is sent to the fog layer to be processed. The coverage zones of some radios with processing units can cross over [11]. As a result, even though they may be found inside the coverage area of multiple nodes, each IoT device is only linked to a single fog node. Compared to cloud nodes, fog nodes are severely lacking in the areas of networking, storage space, and computational power.

The fog layer is a supplementary intermediate layer that processes and analyzes data in real-time and is located close to the end users. The top level of the proposed architecture for a system of health management is made up of a cloud server and a gateway. Extra data processing and storage capacity are often handled by the cloud. Connecting the cloud server and the fog layer, a gateway facilitates bidirectional data transfer. After processing the data from the sensors, fog nodes upload the health condition of a patient to a cloud server with a persistent database. The data is also available whenever needed via cloud storage.

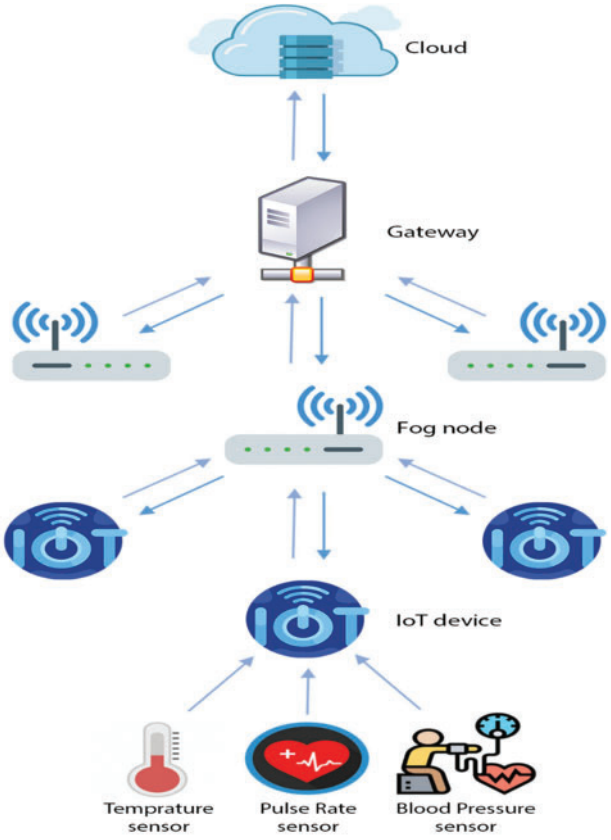

Figure 1: System architecture

4 Load Balancing

In our suggested architecture, fog nodes function similarly to mobile networks. The coverage zones of neighboring nodes may overlap, allowing Internet of Things devices in overlapping locations to connect to both nodes. However, if a node is overburdened with traffic, it may constitute a communication bottleneck, showing that the latency of dataflow is mostly to blame for the delay in the overall response. The data flow's latency on IoT devices is comprised of the aforementioned latency of communication due to traffic load and processing latency due to computing load [11]. As a result, the load balancing solution needs to account for both the data transfer needs and processing capabilities of fog nodes. When one fog node is weighed down by an excessive number of connected devices, it can cause network congestion. Therefore, the extra load is distributed to other, less strained fog nodes. As a result, the previously assigned node will experience less traffic stress, but the newly

assigned node may experience more computational load. Then, certain IoT devices are offloaded to nearby nodes to decrease this computational strain. In this way, the balancing of the compute load on the previous node may cause the newly assigned node's traffic burden to increase. To reduce system-wide latency and network consumption, we need a load-balancing mechanism that proportionately distributes traffic and processing needs among fog nodes. In this paper, we present a SPEA2 Load Balancing (SLB) technique for reducing latency and network use in fog networks, taking into account both communication and compute latencies.

We utilized the same methods as researchers in [11] when proposing the SLB method for measuring the fog network's communication and compute latencies. Eq. (1) through 10 are therefore derived from [11]. Table 1 displays the symbols and their definitions used in this paper. The number of fog nodes established in a certain region can be stated as:

$J = j_1, j_2, j_3, ..., j_n$

Table 1: Key symbols used in this paper

Symbol	Interpretation
J	Set of base stations or fog nodes
$\mathcal{D}$	Set of IoT devices
P(x)	Transmission power of IoT device at location x
g(x)	Channel gain of IoT device at location x
fl(x)	Flow arrival rate at location x
l(x)	Traffic size at location x
$C_j(x)$	Capacity of IoT device at location x
V(x)	Computing size of data flow at location x
C_j	Computing capacity of fog node j
$e_j(x)$	Traffic load density of IoT device at location x
$\hat{e}_j(x)$	Computing load density of IoT device at location x
TL_j	Traffic load of base station j
CL_j	Computing load of base station j
L_m	Communication latency ratio
L_p	Computing latency ratio

Let's pretend that point x in the traffic load area A has an Internet of Things deployment with channel gain g(x), transmission power P(x), and noise power denoted by σ^2. Eq. (1) can be used to determine the signal-to-noise ratio SNR(x) of any given IoT node in a network:

$$SNR(x) = \frac{P(x) \times g(x)}{\sigma^2} \tag{1}$$

The channel gain g(x) is determined as follows using Eq. (2):

$$g(x) = 10 \log_{10} \left[\frac{\lambda^2}{(4\pi d)^2} \right] \tag{2}$$

Note that λ is the wavelength, whereas d is the distance between the IoT device and the fog node. Calculating wavelength λ involves dividing the speed of light by the carrier frequency. If a given Internet

of Things device $C_j(x)$ is linked to a given network node j, and if node j has bandwidth BW_j, then we can write this as an equation:

$$C_j(x) = BW_j \times \log_{10}(1 + SNR(x)) \tag{3}$$

The Internet of Things device has a certain traffic load density for node j at location x. Eq. (4) can be used to represent this traffic load density:

$$e_j(x) = \frac{fl(x) \times l(x) \times b_j(x)}{c_j(x)} \tag{4}$$

According to [12], IoT data flows are governed by the Poisson Point Process, with an average flow rate defined by $fl(x)$, where l(x) is the flow's traffic size and $b_j(x)$ is a binary indicator indicating whether or not the device is linked to the proper node. Using the densities of traffic load of IoT devices and Eq. (5), the traffic load of a fog node can be determined.

$$TL_j = \sum\nolimits_{x \in A} e_j(x) \tag{5}$$

Eq. (6) can be used to determine node j's communication delay ratio L_m, which is based on the assumptions of [12]:

$$L_m(j) = \frac{TL_j}{1 - TL_j} \tag{6}$$

The delay of data flows is observed to be influenced by the computation latency of fog nodes responsible for the computing load. Using Eq. (7) and the assumption that V(x) is the typical size of computation of data flow, we can determine the density of computing load of an IoT device.

$$\widehat{e}_j(x) = \frac{fl(x) \times V(x) \times b_j(x)}{C_j} \tag{7}$$

where C_j is the computing power of the jth cloud node. Computing load at node j can be determined by summing the densities of all the connected IoT devices, as seen in Eq. (8).

$$CL_j = \sum\nolimits_{x \in A} \widehat{e}_j(x) \tag{8}$$

According to the presumptions in [11], it is possible to calculate the computing latency ratio L_p of BS j using Eq. (9).

$$L_p(j) = \frac{CL_j}{1 - CL_j} \tag{9}$$

Using Eq. (10), the authors of [35] aggregated the communication and computation latency of all fog nodes. Then use L as their optimization criterion.

$$L = \sum\nolimits_{j \in J} \left[L_m(j) + L_p(j) \right] \tag{10}$$

These two types of delay have distinct meanings; hence they cannot be added together. Instead of the strategy provided in [34], we employ a multi-objective algorithm to simultaneously optimize both latencies.

4.1 Genetic Algorithm for Load Balancing

Genetic Algorithm (GA) is a heuristic search inspired by Charles Darwin's theory of natural evolution. This algorithm represents the process of natural selection where the fittest individuals are selected for reproduction to generate the next generation of offspring. The common GA algorithm

is used for single-objective problems, but in our work, we have multi-objective optimization. The enhanced power in the complexity of dimension processing in multi-objective situations led us to use the Strength Pareto evolutionary algorithm (SPEA2). The SPEA optimizes for many goals at once, making it a variant of the Genetic Algorithm. An external database of previously found non-dominated solutions is available in SPEA2. At the end of each generation, it is refreshed and a strength value is calculated for each solution [22]. The non-dominated set is archived independently from the population of candidate solutions used in the evolutionary process, representing a form of elitism. SPEA2 is an updated version of SPEA that fixes some of its problems [4]. It features a new archive truncation method, more accurate search help, and a better fitness assignment system. When determining a solution's raw fitness, SPEA2 takes into account both the number of dominant and dominated solutions. This is done to prevent situations in which population members dominated by the same archive members have the same fitness value.

In multiple objective genetic algorithms (MOGA), an individual designated by a chromosome represents a solution for load balancing. A 1-dimensional array corresponding to n genes is used to encode chromosomes. The sequence number of the gene corresponds to the sequence number of IoT devices; each gene has a value of an integer j in the range [1: J], where J is the number of fog nodes, indicating that the associated IoT device is allocated to node j. An example of a load-balancing solution for 10 IoT devices in a cloud-fog system with 3 nodes is: chromosome = [3, 1, 2, 1, 2, 2, 3, 1, 3]. Node 1 conducts the request for IoT numbers 2, 5, and 9, Node 2 is responsible for the IoT set 4, 6, and 7, and Node 3 is responsible for the IoT set 1, 3, 8, 10. Fig. 2 shows the structure of an instance chromosome in the proposed method.

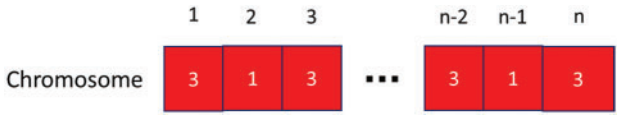

Figure 2: Structure of an instance chromosome

This chromosome encoding method is used due to its adaptability in executing genetic operations, like mutation and crossover, to generate new individuals exploring search space of the solution while inheriting quality gene segments from their parents. In the proposed method, a single-point crossover operation is used, an example of which is shown in Fig. 3. Also, an example of the mutation operation is shown in Fig. 4. In the proposed method, some genes of each chromosome are changed in the mutation operator with the probability Pm. The purpose of this is to keep the members of a population as different as possible and avoid premature and incomplete convergence. Using the jump operator causes the search space to be searched randomly.

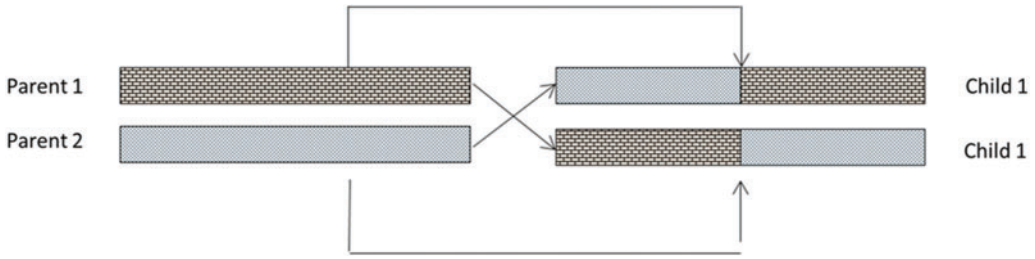

Figure 3: An example of a single-point crossover operation applied in the proposed method

Figure 4: An example of a mutation operation

The initial population consists of all individuals utilized by MOGA to determine the optimal solution. Assuming that the population size is N. These N people are initialized randomly to discover many regions in the search space and to assure the variety of the population in the first generation. People are selected from the initial population and some procedures are performed on them to create the next generation. In our work, we create 200 children as an initial population.

The fitness function quantifies an individual's supremacy in the population. People that have a high value when they comes to fitness symbolize a solution of high quality. In multi-objective optimization, there is generally no practical solution that concurrently minimizes all objective functions. Therefore, special consideration is given to Pareto-optimal solutions, namely, solutions that cannot be improved in any of the objectives without worsening at least one of the other objectives. In mathematical jargon, a viable solution $x_1 \in X$ is said to (Pareto) dominate another solution $x_2 \in X$, if it is more favorable than the latter.

$\forall i \in \{1, \ldots, k\}, f_i(x_1) \leq f_i(x_2)$ *and*

$$\exists i \in \{1, \ldots, k\}, f_i(x_1) < f_i(x_2) \tag{11}$$

A solution $x^* \in X$ (and the accompanying outcome $f(x^*)$) is considered Pareto optimum if there is no other solution that is more advantageous.

As stated previously, we aim to minimize latency and distribute the load evenly. Our proposed fitness function that satisfies specified requirements incorporates variance and the sum of latencies. Consequently, our fitness function will be:

$$f_1 = \sum_{j \in J} L_m(j) \tag{12}$$

$$f_2 = \underset{j \in J}{var}(L_m(j)) \tag{13}$$

$$f_3 = \sum_{j \in J} L_p(j) \tag{14}$$

$$f_4 = \underset{j \in J}{var}(L_p(j)) \tag{15}$$

Consequently, we have four objectives.

The one-point crossover operation is selected through chromosome encoding as an array of integers to produce kids with inherited parental excellent genes. In this procedure, a random crossover point is chosen, the first gene segment is exchanged with the second parent by the first parent, and the remaining genes that remainstay unaltered to generate a new individual. In the process of crossing over a population, parental selection has a significant impact on the algorithm's performance. Each individual possesses an α crossover rate. We set α to 0.8. Each individual in the population is assumed to be the first parent with probability α, and then the Roulette wheel method is used to choose the second parent to participate in the crossover process. With this selection method, high-quality people with a greater value when it comes to fitness have a greater chance of being chosen as parents, guaranteeing that good gene segments are most likely to be passed on to the next generation.

After the crossover procedure, each child undergoes a one-point mutation at a rate of γ. We set γ to 0.1. The position of the mutant gene was chosen at random and substituted with a new value, so assigning the IoT device to be processed at a different node. Mutation improves the limits of the crossover operator by locating the optimal solution when individuals are violent around the extremes or fleeing from the local extremes to explore other regions of the solution space.

Since MOGA is an iterative algorithm, we need to set a termination criterion to complete the simulation. We chose 5000 iterations as the stopping point for the MOGA algorithm. we can explain SPEA2 work algorithm below:

Algorithm: SPEA2 and MOGA for Load Balancing

```
Begin
Initialize Population P
        Evaluate Objective
        Create External Archive A
        For i = 1 to Number of Generations Do
            Compute Fitness of Individual in P and A
            Add Non-dominated Individual from P and A
          If Capacity of A is Exceed Then
            Remove Individual from A by Truncation
              Operator
            End If
          Perform Binary Selection to create Mating pool
         perform crossover
         perform Mutation
    End For
End
```

The algorithm's objective is to locate and maintain a collection of Pareto-optimal solutions, or a front of non-dominated solutions. This is done to prevent situations in which population members dominated by the same archive members have the same fitness value. The algorithm's objective is to locate and maintain a collection of Pareto-optimal solutions, or a front of non-dominated solutions.

5 Experimental Setup

This section details the simulation setup that was used to test the efficacy of the proposed method. The sensors continuously update the fog nodes with data on vital signs like core body temperature, blood pressure, and heart rate. The fog nodes perform the necessary data processing and analysis to determine the severity of a patient's condition. These fog nodes collect the data and send it to the cloud, where it can later be accessed by the patient's IoT device. A gateway is used to relay communications between the cloud server and the fog nodes. We simulated and analyzed our suggested process using iFogSim, a free and open-source toolbox. The most effective simulator for programs that support fog computing is regarded as iFogSim [34]. iFogSim was also used by several studies [23–26,30,31] to model their suggested architectures. A 3000 m by 2000 m area is randomly assigned six fog nodes to be there. Each fog node starts with six connected IoT devices in a 500-meter circle.

In Fig. 5, Fog nodes may overlap in coverage areas because of their haphazard deployment. Each fog node is given a fake Internet of Things device for testing purposes. Using coordinate values, these Internet of Things gadgets are dispersed at random throughout the node's coverage area. The Client Module is installed in IoT devices to collect data from sensors, while the Processing Module is constructed on fog nodes to analyze incoming data and identify the patient's health state. The related IoT device receives the results from the fog node and displays them. Numerous variables, such as CPU length, RAM, bandwidth, etc., must be supplied when generating fog devices in iFogSim. The device-specific iFogSim configuration parameters are presented in Table 2. The term "fog devices" refers to all computational devices produced by iFogSim. However, there are different tiers of computational equipment. A cloud server serves as Level 0's parent node. The Level 1 gateway links the fog nodes to the cloud host. When operating at Level 2, fog nodes are moved closer to the end user, increasing both their processing and storage speeds.

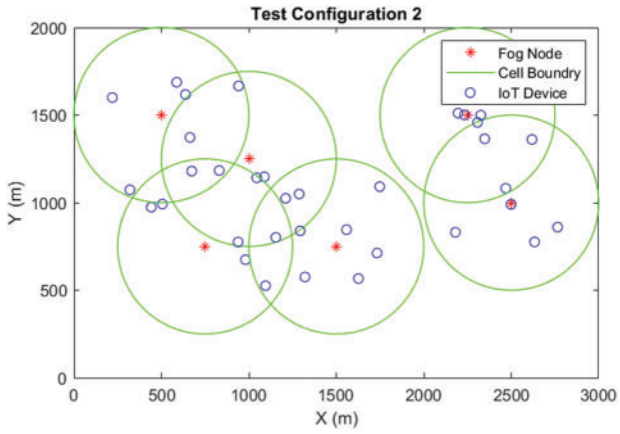

Figure 5: Example of system configuration

Table 2: Values for the fog-based health monitoring system's parameters

Parameter	Cloud	Gateway	Fog	IoT device
CPU length (MIPS)	44800	40000	30000	20000
RAM (MB)	40000	4000	4000	4000

(Continued)

Table 2 (continued)

Parameter	Cloud	Gateway	Fog	IoT device
Uplink bandwidth (MB)	10000	10000	10000	10000
Downlink bandwidth (MB)	10000	10000	10000	10000

Level 3 IoT devices include sensors and actuators. The simulations were performed on a laptop (Intel Core i7 8550u processor and 16 GB of RAM) utilizing the MATLAB 2022a program. We ran several topological simulations of the suggested architecture using iFogSim. To adapt to newer topology configurations, the number of IoT devices per fog node has been steadily rising. Originally, we had four IoT devices connected to each fog node, but as the network expanded, we added more devices. Table 3 displays the fog network topology configurations that were employed in the simulations.

Table 3: iFogSim simulation scenario topologies

Configurations	No. of IoT devices	IoT devices in fog network
Config-1	4	24
Config-2	6	36
Config-3	8	48
Config-4	10	60
Config-5	12	72

Latency and network use are the performance parameters examined when evaluating the suggested implementation. A fog node's traffic and computing load will eventually rise as the number of IoT devices connected to it rises, which will raise the node's network use and latency. In a cloud-only arrangement, a router connects multiple IoT layer devices to the cloud server. Sensor-generated data values are transmitted by IoT devices to the cloud for analysis, and the analyzed data values are then displayed on the IoT device. the topology configuration's size will be gradually increased so that its impact on latency and network use may be studied. In a cloud-only implementation, the setup parameters are shown in Table 4 whereas the values of extra SLB parameters are shown in Table 5.

Table 4: Value of parameters for cloud-only health monitoring system

Parameters	Cloud	Gateway
CPU length (MIPS)	44800	2800
RAM (MB)	40000	4000
Uplink bandwidth (MB)	10000	10000
Downlink bandwidth (MB)	10000	10000

Table 5: Value of parameters used in proposed approach LBS

Parameters	Values
fl(x)	0.50 flows/second
l(x)	0.05 Mbits
V(x)	5000 CPU cycles
P(x)	100 mW
C_j	7.0×10^6
Uplink frequency BW	10 MHz
Carrier frequency	2110 MHz
Noise power level	−104 dBm

6 Results and Discussions

As illustrated in Fig. 6, when the proposed approach is applied to the example in Fig. 7, the load is uniformly distributed.

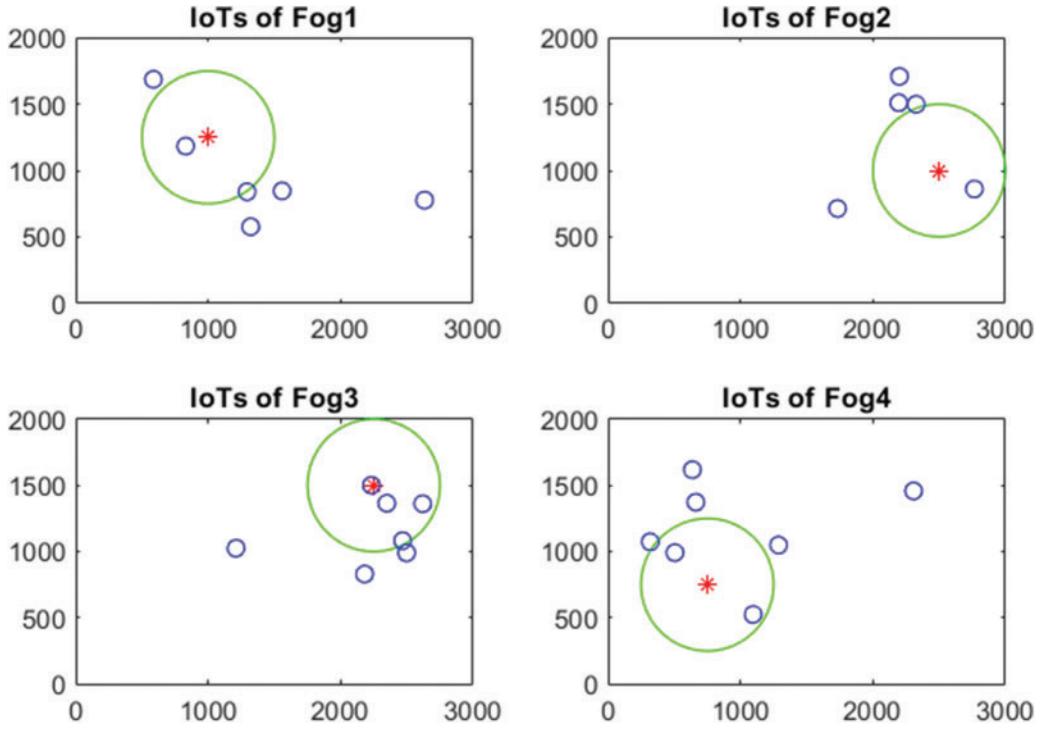

Figure 6: (Continued)

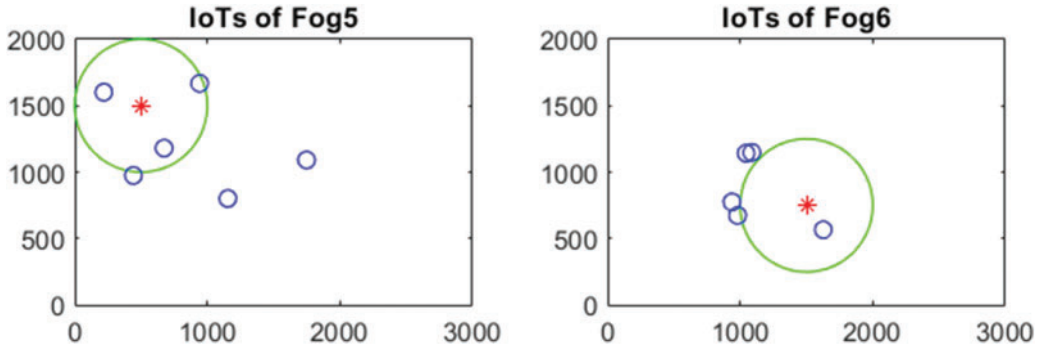

Figure 6: Load balancing result

Each node has between five and seven connected IoT devices. This confirms that none of the fog nodes have been overburdened and that our method is functioning properly. Fig. 7 depicts the overall latency recorded for each fog during iterations to validate the latency criterion.

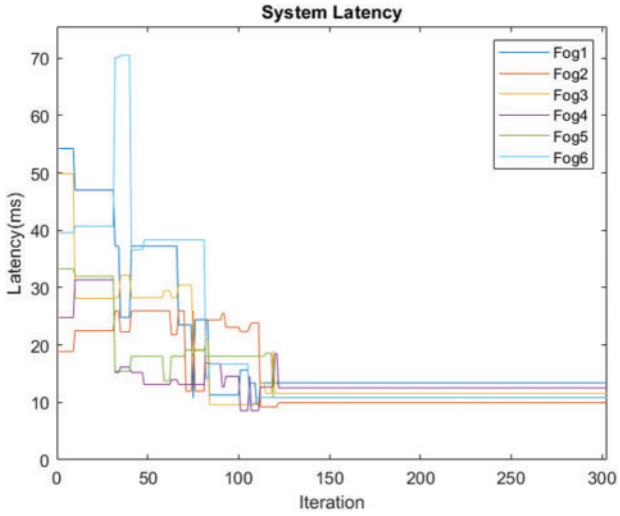

Figure 7: Latency of fog nodes during optimization

After 120 iterations that take less than 100 milliseconds, the algorithm has found the optimal answer, as seen. Fig. 8 balancing method is optimal since latency for the fog nodes has converged, and each fog has a comparable computational load.

Comparing the SLB's performance to that of the cloud-only implementation, FNPA [31], LAB [11], and LBS Scheme [34] are provided here. Tables 6 and 7 display the outcomes of the latency and network utilization performance metrics, respectively. The results of the simulations show that LBS is superior to cloud-only implementations, FNPA, and LAB schemes in the matter of latency reduction. Furthermore, the results demonstrate that LBS drastically reduces network utilization in comparison to cloud-only solutions. In comparison to the LBS, network usage is considerably improved by the FNPA and the LAB Scheme.

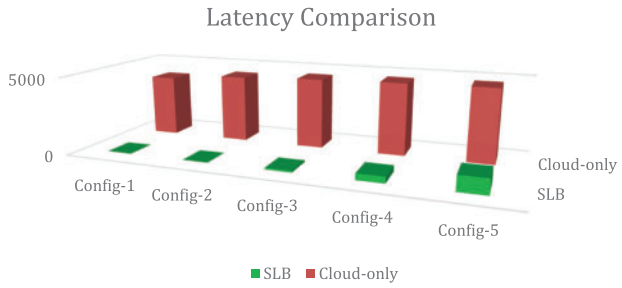

Figure 8: Comparison of latency

Table 6: Comparison of latency (ms)

	SLB	LBS	FNPA	Lab Scheme	Cloud-only
Config-1	11.529	12.81	12.57	12.86	4003
Config-2	11.898	13.22	13.41	161	4367
Config-3	108.9	121	228	358	4544
Config-4	406.8	452	508	870	4662
Config-5	973.8	1082	1148	1232	4737

Table 7: Simulation results for network usage (kB)

	SLB	LBS	FNPA	LAB scheme	Cloud-only
Config-1	337501.2	341668	339724	343489	488369
Config-2	365344.5	372605	378605	385639	662651
Config-3	392524.4	413916	414213	432848	680546
Config-4	416771.3	440857	454958	445508	699054
Config-5	439410.8	466012	479232	466711	717441

Time-critical applications, such as health monitoring systems, benefit from decreased latency. In iFogSim, the latency [34], i.e., the time required to perform a single operation from source to destination, may be calculated by adding compute latency to two times the traffic latency as the time required for transmitting and receiving the data. Fig. 8 compares the latency of SLB with cloud-only services.

It is important to keep in mind that latency increases dramatically with the topology size of a cloud-only implementation. This is because increasing the workload and hence the latency is inevitable when all tasks must be conducted in the cloud. When there is an increase in the users' number and there are no longer any suitable fog nodes with sufficient resources, the cloud server will contact users to finish requests. Because of this, latency increases as the demand for cloud servers grows. Using SLB, fog nodes can handle all incoming tasks, and the complexity of computations requiring fog node selection is reduced. Fig. 9 shows that our suggested scheme, SLB, notably decreases latency in comparison with FNPA, LBS, and LAB schemes.

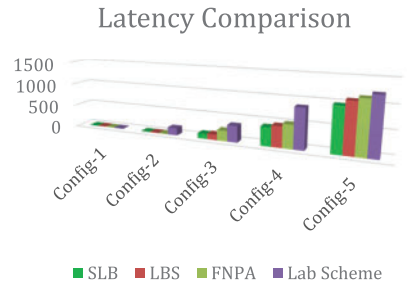

Figure 9: Latency's comparison between fog-based implementations

Fig. 10 compares the simulation results for network utilization in the LAB, LBS, FNPA, the cloud-only, and our suggested scheme SLB. It can be noticed that, for cloud-only implementations, network utilization increases as topology configuration size increases. A router connects each endpoint to the cloud server, ensuring that only one cloud server is handling requests at once and maximizing network efficiency. When using a fog-based approach, there are numerous fog nodes, one of which is solely responsible for the connected IoT devices. This fog node's primary function is to handle requests from the IoT devices it is connected to. When compared to cloud-only systems, the SLB significantly reduces network usage.

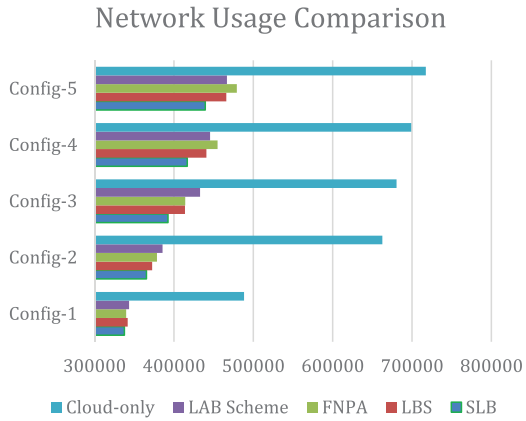

Figure 10: Network usage's comparison

FNPA's simulations show that as the users' number grows, the number of fog nodes capable of handling the demand decreases, and the workload is subsequently shifted to the cloud. As a result, network usage increases in comparison to a fog-based architecture but stays far lower than in a cloud-based implementation. In comparison to cloud-based implementations, the LAB Scheme significantly reduces network utilization by allowing all processing to be done on fog nodes. In our proposed method, computation and analysis are mostly carried out on fog servers, similar to LBS, FNPA, and LAB schemes. From the data in Table 7, it is clear that compared to FNPA, LBS, and LAB schemes, our proposed solution SLB significantly reduces the amount of data transmitted over the network. SLB was effective for use in a comprehensive health monitoring system, despite the insignificant change.

In terms of latency and network utilization performance metrics, our experiments reveal that implementing fog computing is a viable solution for health monitoring systems.

7 Performance Comparison

In this part, a comparison was made with a recently published article related to the topic in Table 8. The comparison relied on a working model, accuracy, energy consumption, network consumption, implementation time, production rate, response speed, and cost.

Table 8: Comparison table

Reference	Work modeling	Accuracy	Energy consumption	Network consumption	Execution time	Latency	Cost
[8]	UT-GATE	High	High	High	High	Low	Low
[9]	IoT-fog-cloud	Low	High	High	Low	High	Low
[13]	ECG-based	High	High	Low	Low	High	Yes
[15]	RUS,FogIoHT	High	High	High	High	Low	High
[19]	HealthFog	Low	High	Low	High	High	High
[20]	Fog computing	Low	High	Low	High	Low	Low
[21]	Healthcare system	High	High	Low	Low	Low	High
[22]	Health monitoring	High	High	Low	High	High	High
[25]	Fog-Cloud-IoT, GPDR	High	High	High	Low	Low	High
[26]	J48Graft	Low	High	High	Low	Low	Low
[28]	LORA	Low	High	Low	Low	High	Low
[29]	IoT, Fog	High	High	Low	High	Low	Low
[30]	Fog, LB and cloud	Low	High	Low	High	High	High
[34]	RTES	Low	High	High	Low	High	Low
[35]	IoHT applications	High	Low	Low	High	Low	High
Proposed work	SPEA2, LAB Scheme	High	Low	Low	High	High	Low

8 Conclusion and Future Work

In this paper, increasing latency and excessive bandwidth consumption were discussed. Timely retrieval of data on a patient and a decrease in the required time for processing data and assessing the severity of a health condition are two benefits of adopting an architecture that is fog-based for an application that is time-sensitive such as a system for monitoring health. Fog computing can be employed when processing data quickly is crucial., The fog-based approach has shown to be more practical in time-critical situations due to its low latency and minimal network utilization.

Our research suggests a fog-based health monitoring system and the SLB load-balancing algorithm. We employ a method that selects a suitable fog node to host the requests that are coming in to fairly spread the load across the fog nodes. Through the use of simulations, the efficacy of SLB with cloud-only, FNPA, LBS, and LAB schemes was evaluated. The results of our simulations demonstrated that our suggested method has superior latency and network consumption compared to both cloud-only and other fog-based implementations, like LAB and FNPA Scheme. The suggested

algorithm can be researched, reviewed, and tested on bigger and more diverse data sets in the future, allowing its utility for other vital signs or the diagnosis of a particular condition to be proven.

In the future, we may also utilize a queuing method to add realism to the current simulation and minimize delay, response time, latency, and energy.

Acknowledgement: The authors like to acknowledge the Department of Computer Science at University of Tabriz for their technical support.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: HMS and JK organized the research. HMS and JK designed the model and the computational framework. HMS carried out the implementation. HMS and JK wrote the manuscript. JR critically revised the manuscript. All authors read the manuscript and confirmed its final version.

Availability of Data and Materials: None.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Alshamrani, "IoT and artificial intelligence implementations for remote healthcare monitoring systems: A survey," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 4687–4701, 2022.

[2] V. Khanh Quy, N. van Hau, D. van Anh and L. Anh Ngoc, "Smart healthcare IoT applications based on fog computing: Architecture, applications and challenges," *Complex & Intelligent Systems*, vol. 8, no. 5, pp. 3805–3815, 2022.

[3] C. Butpheng, K. H. Yeh and H. Xiong, "Security and privacy in IoT-cloud-based e-health systems-A comprehensive review," *Symmetry*, vol. 12, no. 7, pp. 1191, 2020.

[4] A. Katal, "Leveraging Fog Computing for Healthcare," in *Deep Learning Technologies for the Sustainable Development Goals: Issues and Solutions in the Post-COVID Era*, Springer Nature, pp. 51–68, 2023.

[5] Y. Y. Ghadi, I. Akhter, S. A. Alsuhibany, T. A. Shloul, A. Jalal *et al.,* "Multiple events detections using context-intelligence features," *Intelligent Automation & Soft Computing*, vol. 34, no. 3, pp. 1455–1471, 2022.

[6] S. Kumar, P. Tiwari and M. Zymbler, "Internet of Things is a revolutionary approach for future technology enhancement: A review," *Journal of Big Data*, vol. 6, no. 1, pp. 1–21, 2019.

[7] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi *et al.,* "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[8] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira *et al.,* "The internet of things, fog and cloud continuum: Integration and challenges," *Internet Things*, vol. 3–4, pp. 134–155, 2018.

[9] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gaoand *et al.,* "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[10] B. Farahani, F. Firouzi, V. Chang, M. Badaroglu, N. Constantand *et al.,* "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare," *Future Generation Computer Systems*, vol. 78, pp. 659–676, 2018.

[11] M. Mukherjee, L. Shu and D. Wang, "Survey of fog computing: Fundamental, network applications and research challenges," *IEEE Communications Surveys and Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[12] B. Farahani, M. Barzegari, F. Shams Aliee and K. A. Shaik, "Towards collaborative intelligent IoT eHealth: From device to fog and cloud," *Microprocessors and Microsystems*, vol. 72, no. 7, pp. 102938, 2020.

[13] S. Kunal, A. Saha and R. Amin, "An overview of cloud-fog computing: Architectures, applications with security challenges," *Security and Privacy*, vol. 2, no. 4, pp. e72, 2019.

[14] A. Mukherjee, D. De and S. K. Ghosh, "FogIoHT: A weighted majority game theory based energy-efficient delay-sensitive fog network for Internet of health things," *Internet of Things*, vol. 11, no. 8, pp. 100181, 2020.
[15] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. of IFIP/IEEE Symp. on Integrated Network and Service Management (IM)*, Lisbon, Portugal, IEEE, pp. 1222–1228, 2017.
[16] R. A. C. da Silva and N. L. S. da Fonseca, "On the location of fog nodes in fog-cloud infrastructures," *Sensors*, vol. 19, no. 11, pp. 2445, 2019.
[17] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered IoT," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 253–262, 2020.
[18] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya *et al.*, " HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments," *Future Generation Computer Systems*, vol. 104, pp. 187–200, 2020.
[19] P. H. Vilela, J. J. P. C. Rodrigues, P. Solic, K. Saleem and V. Furtado, "Performance evaluation of a fog-assisted IoT solution for e-Health applications," *Future Generation Computer Systems*, vol. 97, pp. 379–386, 2019.
[20] A. Mukherjee, S. Ghosh, A. Behere, S. K. Ghosh and R. Buyya, "Internet of Health Things (IoHT) for personalized health care using integrated edge-fog-cloud network," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1–17, 2020.
[21] H. Ben Hassen, W. Dghais and B. Hamdi, "An E-health system for monitoring elderly health based on Internet of Things and fog computing," *Health Information Science and Systems*, vol. 7, no. 1, pp. 1–9, 2019.
[22] E. Badidi and K. Moumane, "Enhancing the processing of healthcare data streams using fog computing," in *Proc. IEEE Symp. on Computers and Communications (ISCC)*, Barcelona, Spain, IEEE, pp. 1113–1118, 2019.
[23] H. Saidi, N. Labraoui, A. A. A. Ari and D. Bouida, "Remote health monitoring system of elderly based on Fog to Cloud (F2C) computing," in *Int. Conf. on Intelligent Systems and Computer Vision*, Fez, Morocco, IEEE, pp. 1–7, 2020.
[24] O. Debauche, S. Mahmoudi, P. Manneback and A. Assila, "Fog IoT for health: A new architecture for patients and elderly monitoring," *Procedia Computer Science*, vol. 160, pp. 289–297, 2019.
[25] M. Devarajan, V. Subramaniyaswamy, V. Vijayakumar and L. Ravi, "Fog-assisted personalized healthcare-support system for remote patients with diabetes," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 10, pp. 3747–3760, 2019.
[26] T. Nguyen Gia, I. B. Dhaou, M. Ali, A. M. Rahmani, T. Westerlund *et al.,* "Energy effcient fog-assisted IoT system for monitoring diabetic patients with cardiovascular disease," *Future Generation Computer Systems*, vol. 93, pp. 198–211, 2019.
[27] S. S. Vedaei, A. Fotovvat, M. R. Mohebbian, G. M. E. Rahman, K. A. Wahid *et al.,* "COVID-SAFE: An IoT-based system for automated health monitoring and surveillance in post-pandemic life," *IEEE Access*, vol. 8, pp. 188538–188551, 2020.
[28] H. Ben Hassen, N. Ayari and B. Hamdi, "A home hospitalization system based on the Internet of Things, fog computing and cloud computing," *Informatics in Medicine Unlocked*, vol. 20, pp. 100368, 2020.
[29] H. A. Khattak, H. Arshad, S. U. Islam, G. Ahmed, S. Jabbar *et al.,* "Utilization and load balancing in fog servers for health applications," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–12, 2019.
[30] A. Paul, H. Pinjari, W. H. Hong, H. C. Seo and S. Rho, "Fog computing-based IoT for health monitoring system," *Journal of Sensors*, vol. 2018, pp. 1–7, 2018.
[31] R. M. Abdelmoneem, A. Benslimane, E. Shaaban, S. Abdelhamid and S. Ghoneim, "A cloud-fog based architecture for IoT applications dedicated to healthcare," in *Int. Conf. on Communications*, Shanghai, China, IEEE, pp. 1–6, 2019.
[32] S. R. Hassan, I. Ahmad, S. Ahmad, A. Alfaify and M. Shafiq, "Remote pain monitoring using fog computing for e-healthcare: An efficient architecture," *Sensors*, vol. 20, no. 22, pp. 6574, 2020.

[33] M. Verma, N. Bhardwaj and A. K. Yadav, "Real time efficient scheduling algorithm for load balancing in fog computing environment," *International Journal of Information Technology and Computer Science*, vol. 8, no. 4, pp. 1–10, 2016.
[34] E. C. P. Neto, G. Callou and F. Aires, "An algorithm to optimize the load distribution of fog environments," in *IEEE Int. Conf. on Systems, Man, Cybern (SMC)*, Banff, AB, Canada, IEEE, pp. 1292–1297, 2017.
[35] A. Hazra, P. Rana, M. Adhikari and T. Amgoth, "Fog computing for next-generation internet of things: Fundamental, state-of-the-art and research challenges," *Computer Science Review*, vol. 48, pp. 100549, 2023.