Fog Computing Performance Optimization in Healthcare Environment

Ameenah Mohammad Alharbi[1] and Mohammed Abdullah Al-Hagery[1]
[1]Department of Computer Science College of Computer
Qassim University Buraydah 51452 Saudi Arabia

Article Info
Page Number: 01 – 23
Publication Issue:
Vol 72 No. 2 (2023)

Abstract: Cloud computing is very powerful and efficient technology, but it suffers from some efficiency issues due to its distributed nature. With the advent of fog, the large cloud is divided into many small cloudlets. Fog computing is one of the latest techniques in today's cloud computing environment. Each cloudlet is a small and new architectural element extending today's cloud computing infrastructure. The term" cloudlets" has been combined with fog computing, as it enhances the characteristics of cloud computing. Today, fog computing represents a basic layer and an essential cloud component. Consequently, this technology can help solve efficiency issues. Therefore, this paper aims to improve the efficiency of the cloud computing environment by optimizing the efficiency parameters results of fog computing. The parameters include response time, latency, and network utilization. The solution is to design a simulation system to minimize the redundancy of tasks and increase the performance rate. The results show an improvement in the performance of the proposed system from 40 to 48% based on the three parameters (response time, network utilization, and latency). Therefore, these results are promising and very significant, which encourages the applying the proposed system in the real healthcare environment.

Article History
Article Received: *15 February 2023*
Revised: *20 April 2023*
Accepted*: 10 May 2023*

Keywords: - Cloud Computing; Fog Computing; Performance Parameters; Response Time; Network Utilization; Latency; Healthcare.

1. Introduction

Fog computing is a new framework for the geographic distribution of computing resources. Fog computing is similar to cloud computing. It serves as a gateway to computing resources and uses similar service concepts as cloud computing. Unlike cloud computing, which focuses on a few high-performance data centers, fog Computing relies on many highly dynamic and heterogeneous resources with reasonable capacity, called fog nodes. The main advantage of fog computing over cloud computing is its proximity to end devices such as sensors, actuators, smartphones, smart cameras, Internet of Things (IoT) devices, etc. It manages and utilizes resources effectively [1].

IoT continues to evolve in parallel with advances in relevant technologies. The term"IoT" has gained popularity and is now commonly used. The United States, the European Union, China, Japan, and Korea have introduced IoT development projects nationally. Many researchers have noted that the IoT contains a growing number of new technologies and has become a necessity in the life of society [2]. IoT architectures have been thoroughly researched to serve mass applications [3]. The main reason for separating data processing from applications is that the cloud computing service often appears as a third party. In reality, massive data processing remains an IoT bottleneck. Some researchers have suggested for years that cloud computing could solve this problem and have proposed numerous cloud-based IoT architectures [4], [5]. The significance of our contribution lies in improving the efficiency of cloud computing through fog computing to operate at high efficiency while reducing the resources required performing the tasks. The objective of this paper is to minimize the challenges of cloud computing through fog computing by optimizing the performance values based on the following parameters:

- Response Time
- Network utilization
- Latency

Many experiments were conducted, a simulation system was developed, and a tuning process for the values of the selected parameters was performed by selecting the best scenario for the simulation system to be applied in a real environment. The system consists of three layers; cloud layer, fog layer, edge layer, and cloudSim. After that, the iFogSim was applied in a healthcare case study.

The rest of the paper is organized as follows: section 2 introduces cloud computing and fog computing in the context of healthcare, including basic principles and concepts of fog computing architecture. Whereas section 3 discusses the related works. The research methodology is presented in section 4, which also explains the simulation tools used in the proposed method, while section 5 demonstrates the results and discussion. Section 6 presents the conclusions, and section 7 highlights future work.

2. Cloud computing and fog computing

There is a lack of consensus in standardizing fog computing, cloudlets, edge computing, and other terms have been used to describe fog. Different research groups have proposed many different definitions of fog. Since there is a research gap in definitions and standards for fog computing, this paper uses the definitions of Atlam et al., [6], [7], in which fog computing is defined as a type of distributed computing architecture, which manages a set of application services locally in the network and smart devices. In contrast, others are managed in the cloud [8]. This section highlights some paradigms proposed to bring the cloud closer to the end devices. In addition, it explains the advantages and disadvantages of introducing fog computing as an ideal platform for IoT.

IoT has revolutionized the ever-evolving field of information and communication technology. Today's smart technologies, such as tablets, computers, and smartphones, have changed how machines, sensors, and vehicles are used in a variety of applications [9]. Basir et al. claimed that cloud computing further enhances the user experience in current applications by providing real-time processing, low latency and storage, and high data rate reliability. Nguyen et al. noted that using edge devices in combination with the Multi-Access Edge Computing (MAEC) paradigm gives fog computing a significant advantage in web optimization and ensures reliability in the context of responsive service.

Unlike servers that rely solely on Content Delivery Networks (CDNs), the combination of fog-cloud computing and the MAEC paradigm has improved computing performance. Currently, the new IoT devices are more fixated on using smaller and smaller processors while increasing memory and speed as microprocessor technology continues to grow [10]. According to Basir et al., wireless sensors have promoted the use and application of fog-cloud computing in various applications, such as traffic monitoring, patient state monitoring, etc. Consequently, modern devices' storage capacities have been increased by applying advanced computing techniques enabled by fog nodes [11]. As a result, advances continue to be made in microprocessor technology that uses smaller CPUs and more memory to provide a better user experience. Many research studies focused on fog and cloud computing, botnets, and Distributed Denial of Service (DDoS) that highlight their various technological characteristics [12]. Open edge computing describes edge computing as a data processing paradigm. It offers small data facilities (edge nodes) close to users and is designed to provide an enhanced customer experience by interacting with storage and compute resources just a hop away from the viewer [5]. The open fog consortium notes that fog computing is often mistakenly referred to as edge computing, but there is a big difference between the various terms. Although they share similar concepts, edge computing is limited to edge devices (i.e., in the IoT node network) and excludes the cloud from its architecture [13]. On the other hand, fog computing is a multi-layer network not limited to a local network but offers services everywhere from the cloud to objects [7]. The ”extreme edge” of IoT analytics is pushed to the ”extreme edge” by fog computing [14]. In this new computing paradigm, the fog has spread even further, where the fog computing layer consists of fog nodes, so-called because of its lightweight. It is even closer to the end devices than the fog nodes, which are specialized and dedicated nodes with minimal computing resources (e.g., microcomputers and microcontrollers).

The IoT is a set of interconnected devices connected with a network with sensors to improve the collection and exchange of data with the cloud at the fog computing level. The IoT involves connecting numerous devices to improve data analysis and autonomous decision-making. An IoT system consists of various functional blocks, which provide different functions for the fog computing system. These blocks include communication, devices, management, services, and applications [15].

The devices facilitate the exchange and collection of data from other related devices, such as cloud-based devices. The IoT system provides functions such as data publishing, device control, device recovery, device modeling, and data analytics. The application layer serves as an interface that provides essential modules for monitoring and controlling various aspects of the system [16].

Regarding the IoT, fog computing could be considered the first (optional) layer. The IoT fog-cloud systems' design, development, and operation require significant financial investment and simulators. There are reviews of cloud, IoT, and fog simulators that summarize their fundamental capabilities and compare them based on specific metrics [17]. However, some of these simulators can already examine the behavior of the IoT-fog cloud system to a limited extent. Comparing two cloud simulators based on the intelligent weighting capabilities of Virtual Machine (VM) placement methods was also an inspiration for the research work [18]. This is a similar strategy compared to more advanced versions of these simulators.

A. Cloud Computing Services and Challenges

Cloud computing provides users with numerous computing, communication, and problem-solving services. Google, IBM, Microsoft, and Amazon are examples of the most prominent cloud service providers. Cloud services are typically offered on a pay-as-you-go basis [19]. Consequently, users only pay for their computing resources and data management services.

On the other hand, cloud computing has several characteristics, such as rapid elasticity, large scale, on-demand self-service, high extensibility, ubiquitous network access, and resource pooling. This means that the scope of the cloud can be increased to meet high customer demands [20], [21]. In contrast, cloud computing suffers from many challenges from rapidly growing technology, such as security, confidentiality, integrity, availability, isolation, authentication, identification, and authorization [22].

These challenges lead to data manipulation and dissemination of inaccurate or misleading information to users. On the other hand, authentication and authorization [23]. Due to the technological advancements in today's world, the cloud is still struggling with this difficulty and is developing new solutions every day [24]. For a customer expecting better security in cloud computing, this is disastrous. Another critical challenge with cloud computing is confidentiality. If an attacker is in a virtual machine, it is easy to eavesdrop on the other device, find its address name, and access the customer's data. Data loss is possible, and some solutions for this situation have been presented [25].

B. Features and Architecture of fog computing

Fog computing is consistent with edge computing, where tasks are moved to the edge [26]. In other words, it is an end-to-end architecture that brings network functions closer to users. Fog computing is characterized by its edge location, which allows it to process applications with low latency [27]. Fog computing is characterized by location awareness, which means that the fog node can track end users' devices and infer their location to enable mobility easily. As a result, the relationships between fog and cloud, also between fog and fog, become more important as fog computing becomes more capable of capturing the required data [28]. The architecture of fog computing includes the following:

1. Terminal layer or edge layer: It is a lower layer consisting of IoT devices, sensors, and actuators that communicate directly with the user, where data is transmitted from these devices. The lower-level devices have no processing capabilities.
2. Fog layer: This layer consists of different types of nodes, such as switches, gateways, routers, access points, and fog servers arranged in two layers. The lower-level fog nodes have lower computational capacities than the higher-level fog controller node. The fog controller node controls the entire node cluster.
3. Cloud layer: The nodes in this layer have extensive resources that provide cloud services. The communication latency is very high when the applications are deployed in the cloud layer. More energy is consumed for the cooling system in the cloud. To promote the control functionality of the fog environment. In [29], the authors demonstrate a general system architecture of the fog integrated with the cloud and they highlight the heterogeneously distributed nodes forming a cluster. Each cluster has a control node called the Fog Controller Node (FCN).

1) Simulation and Tools:

Simulator is software that depends on mechanisms with a collection of mathematical equations [30]. Many simulators and tools can simulate such environments and measure various properties such as latency, network utilization, and response time. Some of the simulators and tools intended for fog computing [31] are as follows:

- CloudSim: is the most popular simulation tool for the cloud computing environment, developed by the GRID lab at the College of Melbourne [32], which enables seamless modeling and simulation of cloud computing infrastructure development. Researchers can use this simulator to focus on a specific system design for the cloud system infrastructure.
- iFogSim: is one of the simulations of fog computing environments to evaluate resource management and planning strategies. It works over the closest cloud resources to the end device under different scenarios. It allows designing different architecture scenarios for fog computing and evaluating different aspects such as the network bandwidth [31]. In this simulator, any fog environment infrastructure is simulated by adding fog nodes and edge devices with features such as resource capabilities. A Graphical User Interface (GUI) creates the network topology for building the entire infrastructure in JSON format. This simulator can also measure performance metrics and simulate edge devices [33].
- Cloud analyst: is an open-source simulator obtained from the cloud that allows determining the performance of cloud services. It simulates a geographically distributed environment that is used to evaluate the performance of computing servers [34]. The simulator's configuration parameters can be adjusted as the researcher desires to evaluate their design. Some customized parameters are the number of requests generated per user per hour, the number of nodes, the amount of memory, the number of processors, the number of virtual machines, the network bandwidth, and other necessary parameters.

Identifying an appropriate simulator depends on the analysis processes of the requirements. As seen from the previous description of the three simulators, CloudSim is the simulator specifically for the cloud environment. Its results are displayed in the console, not in a graphical user interface. It also does not fully consider the location of the end device and the data center. In this simulator, proximity is also not considered as a whole. Therefore, clouds would not be preferable. The next simulator is iFogSim, a fog computing simulator designed for resource management and scheduling policy in a fog environment. This simulator has strong points, such as measuring the required performance parameters, and has some limitations, such as unsupported and challenging to understand horizontal communication. The third simulator is a cloud analyst that allows the description of endpoints and data centers. It includes information about the location of users generating tasks, the geographic location of data centers, and the number of endpoints, nodes, and resources in each node. This information can be used to determine performance metrics such as processing time, response time, and response cost.

2) Fog Modeling in iFogSim:

iFogSim is one of the most widely used cloud and fog computing research, both of which use CloudSim-based extensions [31]. The following are the primary physical components of iFogSim [32]:

- A Java-based simulator: sensors that generate data as a tuple to represent information.
- Actuators with a geographic location and gateway connection reference;
- Sensors that generate data in the form of a tuple representing information.

iFogSim's AppModule and AppEdge are the key logical components that attempt to mimic a distributed application. To run an application on a fog device, module mapping, and controller are the two key management components [33]. The request is forwarded to a higher-level object if a fog device is not discovered. Specifying a fog system's physical and logical elements is necessary before defining the control unit itself [34].

3) Literature Comparison of iFogSim:

Based on the review and investigation, it is found that, in general, CloudSim is ineffective in simulation modeling for fog. The iFogSim is considered the most optimal simulation tool due to some features, including; performing hands-off migration, performing live migrations, outsourcing better, having an advanced IoT management option, supporting shutdown, and providing advanced networking options. The most significant advantage is its mobility support, which other simulation tools lack. Consequently, the iFogSim is the best simulating tool for fog using the 3D modeling principle [35].

4) Comparison between the modern Fog and traditional Fog implemented in EHS:

Table 1 presents a comparison applied to the traditional fog having un-optimized features and modern fog having optimized features has a very sharp difference in terms of parameters. The modern fog uses a clustered indexed approach in which the IoT actuators send the response to the IoT devices. The last and final layer is the fog, instead of the cloud layer in modern fog architecture. With the advent of clusters, the downtime has been reduced to a minimum because of the next-hope concept. Where each time the next hope takes the request, whenever the new request is not ready to accept the challenge of IoT actuators. If the clustered approach is not ready to accept the request because of flooding or overloading. The request is moved to the upper layer called the Fog implementation layers which inherits the same properties as the centralized Cloud for processing the request and if there come any issues that cannot fulfilling the request due to the Fog approach busy the request then moved to the centralized Cloud servers leaving all the architecture behind [35].

Table 1. Traditional and Modern Fog

Modern Fog	**Traditional Fog**
Clustered	Non-clustered
No-Cloudlets	Cloudlets
Less processing power	More processing power
Fast response rate	Slow response rate
Increase in 0 downtime	Decrease in 0 downtime
Faster rate	Slow rate
Migrations	Non-Migrations

3. Related Works

Much research has been done to improve the performance of fog computing architecture, resource management, and task scheduling algorithm concepts. For example, Abdulgalil [36] proposed a multi-layer fog computing architecture based on the analysis of Electrocardiography (EEG) signals. The proposed architecture compares three layers. The first layer is responsible for generating data and transmitting it to the second layer. The second layer takes advantage of emerging fog computing technology for lightweight analysis based on ECG signals. The third layer is responsible for comprehensive analysis based on EEG signals and data storage in the cloud. In addition, Abdelaziz [37] presented a new concept of collaboration architecture for fog computing. The proposed architecture represents a reference model to better design and implement fog platforms. The proposed architecture provides a reference model to simplify and make the design and implementation of fog platforms more efficient. Santos [38] designed a multi-layer fog computing architecture consisting of three layers at the edge, near the end device in the cloud. The performance of downloading video from multiple layers in a given geographic area was evaluated with multimedia applications. Salman and Jain [15] introduced an IoT fog cloud framework to reduce service delay for IoT applications. The analytical model was developed to evaluate their policy of the proposed framework, which shows how it helps reduce the IoT service delay. Hao et al. [39] proposed a prototype system in the software architecture of fog computing to integrate various designs of IoT communication devices with fog nodes. The design includes a prototype software architecture showing the evaluation of this system.

To provide a solution for IoT services, Varghese et al. [40] proposed a hybrid fog-cloud architecture for low response applications, such as firefighting, that supports a growing variety of applications, including fifth-generation wireless systems and those in IoT, as well as embedded Artificial Intelligence (AI). In addition, Gupta and Dastjerdi [41] studied a task-offloading strategy to minimize energy consumption in mobile devices and developed a priority-based task scheduling algorithm using an edge server. Execution time and execution cost in task data size and latency requirements are performance metrics. The performance evaluation results show that the proposed algorithm reduces the task completion time and improves the quality of services.

Likewise, Mohan et al. proposed a distributed task processing on the edge of fog-cloud participating in the network to assign the processing tasks to the nodes. They developed the Least Processing Cost First (LPCF) method, which is the optimal processing time [42]. Peixoto et al. [23] also introduced a scheduling algorithm in the hierarchical layer of fog and cloud computing with three-tier architecture. Their work shows the scheduling strategies that can be developed in a fog computing environment. In their research, the scheduler prioritizes cloudlet usage and optimizes other objectives, such as reducing network utilization and cloud cost.

Furthermore, Kafhali and Sala [43] proposed a task scheduling algorithm in fog computing devices based on classification and data mining techniques. They have developed a novel classification algorithm and task-scheduling model based on the Apriori algorithm. The task with the least completion time was selected to be executed at the fog node with the least completion time. Also, Wang et al. [31] proposed task scheduling in the fog computing scenario. This strategy is based on a hybrid heuristic algorithm, which mainly solves the problem of terminals with limited computing resources, making the system suitable for real-time and efficient terminal processing tasks. Moreover, Nguyen et al. [44] introduced a new algorithm to optimize task scheduling problems for a bundle of tasks in a cloud-fog environment regarding execution time and operation cost.

The proposed algorithm can flexibly meet users' requirements for high-performance processing and cost efficiency. Their work is limited because they only tested the algorithm on small data sets. Many algorithms for task scheduling in cloud computing and fog computing are used in the real environment. However, there are many algorithms for task scheduling in fog computing, as shown in Table 2. The two algorithms with a good response time are First Come First Served (FCFS) and Shortest Job First (SJF), where the SJF usually gives better solutions as described in the literature.

In addition, there are other scheduling algorithms, such as the Round Robin (RR) algorithm: In Round Robin scheduling, where the processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or quantum of time. If a process does not complete before its CPU time expires, the CPU is preempted and given to the following process waiting in a queue. The preempted process is then placed at the back of the ready list [45], as well as the min–min algorithm, which chooses small tasks to be executed first, thereby delaying the large tasks for a long time. Likewise, the max-min algorithm chooses large tasks to be executed first, thereby delaying the small tasks for a long time [45], [46].

In some related works, the task scheduling algorithm is used on three-level algorithms that cannot consider the capacity of fog nodes and task priority. In this case, the incoming tasks from the IoT device are directly assigned to the available fog node without considering the node's capacity. Moreover, in the proposed algorithm, the tasks are classified based on their size and scheduled with the appropriate layer of the fog architecture to reduce the response time.

Table 2. Comparison of Task Scheduling Algorithms

Scheduling algorithms	**Description**	**Architecture**	**Parameters**	**Limitations**
FCFS [43].	Tasks were assigned to VM by using the arrival time	Third tiers (IoT device, fog node, cloud node)	Arriving time, resource availability	It doesn't consider the capability of the resource, response time and tasks Priority
SJF [31]	Small tasks are assigned first to VM by considering resource availability	Third tiers (IoT device, fog node, cloud node)	Availability of resources, and data size	It doesn't consider the capability of the resource, response time, and starvation may occur.

The proposed algorithm overcomes the limitations of the existing algorithms, such as the FCFS and SJF. It uses the four-layer architecture and assigns the task to each of the corresponding layers of fog computing. Then it minimizes the starvation problem and reduces response time.

4. The Methodology

The proposed approach is a network composed of cloud nodes, various fog nodes, and edge devices. The exact coordinates are continuously determined for the cloud nodes before the simulation. The fog nodes and edge devices are generated with arbitrary coordinates according to a uniform distribution. Initially, the fog nodes are connected to the cloud nodes by default, regardless of their distances. However, the edge devices will not have any connections with the cloud node or fog nodes at the beginning. After that, the nodes will start connecting the edge devices to the fog nodes when these nodes are within a predefined coverage area assigned to each fog node. The procedures include three scenarios, as follows:

1) When the device is not within the range of any fog node, the application will be sent directly to the cloud. (2) When the device is within only one fog node and then check for:

- If this node has enough resources, then the application as a whole is sent to this node.
- If this node does not contain enough resources required by the device, then the node will start checking for any neighboring fog nodes in its scope, which can be combined to form a neighboring group.
- If no fog nodes are in the scope to form a cluster or the potential neighbor groups that cannot handle the application. The request, in this case, is forwarded to the cloud through the initial node that received the request.

(3) When the machine falls within the range of many fog nodes, then there are three probabilities:

(a) At least one node has enough resources, so the application is sent to the node, giving the shortest response time. None of the nodes can handle the request independently, so it checks if more than one node can handle the application together to form an in-scope collection.

(b) Potential in-scope groups don't have the necessary resources, so then in-scope nodes will start checking neighboring nodes for inclusion to form a new adjacency group.

(c) Potential neighbor groups cannot process the request, so the request is forwarded to the cloud through the node closest to the device. Selecting the most appropriate fog node to form an in-scope or an adjacency group for a request service depends on a semantic description of the nodes' resources and application requirements, e.g., memory, node processing power, number of tasks, task size, and number of task instructions.

The requested tasks' distribution among the cluster's fog nodes is treated as an optimization problem to reduce the number of nodes used and the response time. The steps of the proposed methodology can be summarized as follows:

1. Identify the basic requirements for the proposed system to optimize cloud computing performance.
2. Initiate cloud layer.
3. Initiate the fog layer.
4. Initiate edge layer.
5. Applying optimization algorithm.
6. Check the node's availability.
7. Close resources and provide the final results.

Figure 1 shows the flowchart of the proposed system, which represents the simulation processes to improve the performance of fog computing based on an actual case study. The steps represent all the programming phases the system goes through by simulating each layer to improve the performance and get the results.

The algorithm of the proposed system contains the pseudocode of the simulation processes of improving the performance of fog computing according to the selected case study. At startup, the cloud layer shows the availability of on-demand computing resources, specifically data storage (cloud storage) and computing power.

In large clouds, tasks are often distributed across multiple data centers. Cloud computing takes advantage of resource sharing in healthcare to keep data consistent. Healthcare also has emergency response systems that require real-time operations. A distributed component must be added to the traditional centralized cloud computing architecture.

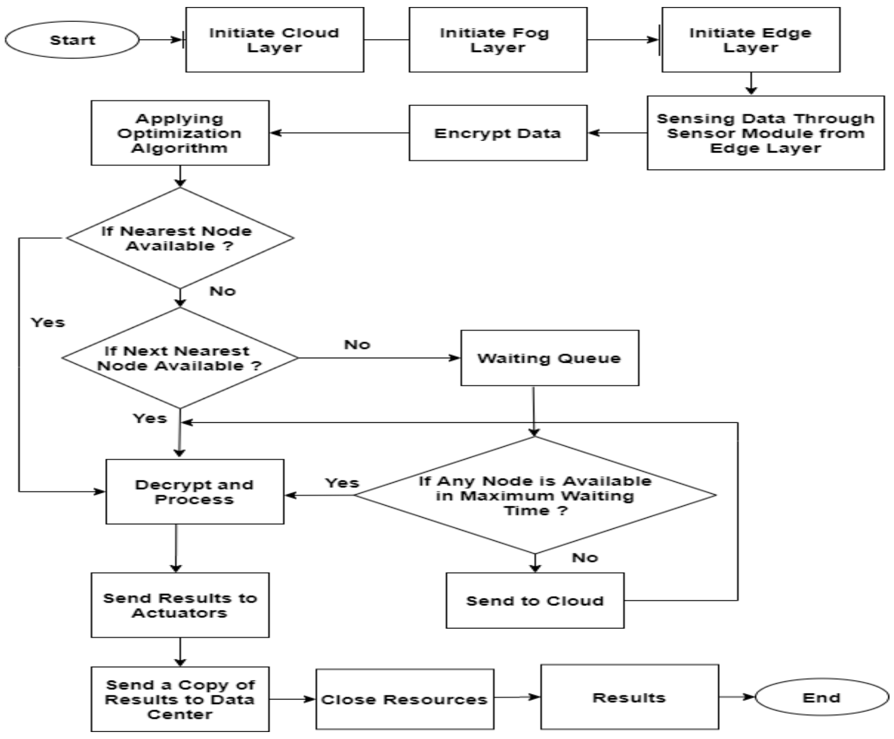
Start
Initiate Cloud Layer
Initiate Fog Layer
Initiate Edge Layer
Sensing Data Through Sensor Module from Edge Layer
Encrypt Data
Applying Optimization Algorithm
If Nearest Node Available ?
No
Yes
If Next Nearest Node Available ?
No
Waiting Queue
Yes
Decrypt and Process
Yes
If Any Node is Available in Maximum Waiting Time ?
No
Send Results to Actuators
Send to Cloud
Send a Copy of Results to Data Center
Close Resources
Results
End

Fig. 1. The flowchart of the proposed system

In a distributed architecture, tasks are split and distributed across multiple nodes. Figure 2 shows the structure of the class diagram of the system. Also, it displays the main classes of iFogSim. The class interaction model shows the components and the relationships between these components of the simulation system.

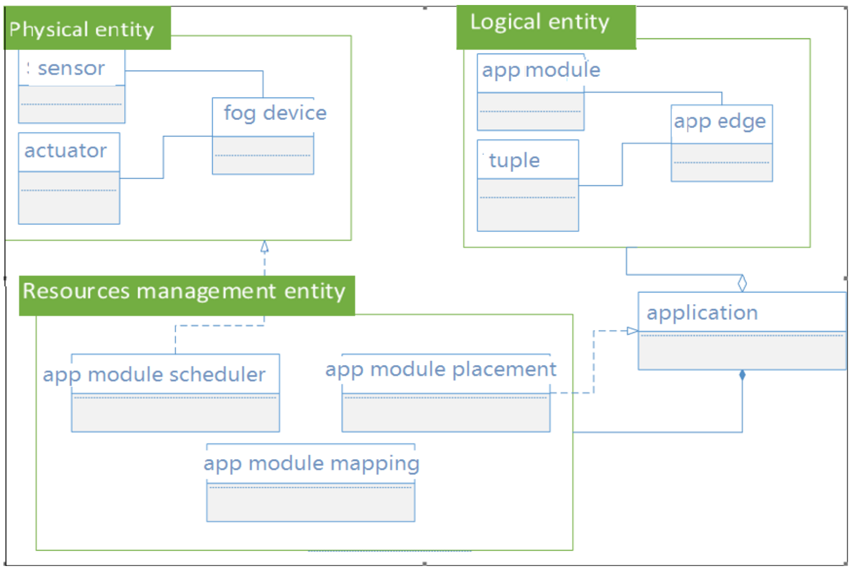

Fig. 2. Main classes of iFogSim.

On the other hand, Figure 3 demonstrates the sequence diagram of the interaction between the sensor and fog device execution. A data tuple is generated by a sensor and sent to the gate to which the sensor is connected. The callback function is used to handle the incoming data tuple that is invoked as soon as the tuple reaches the fog device (input).

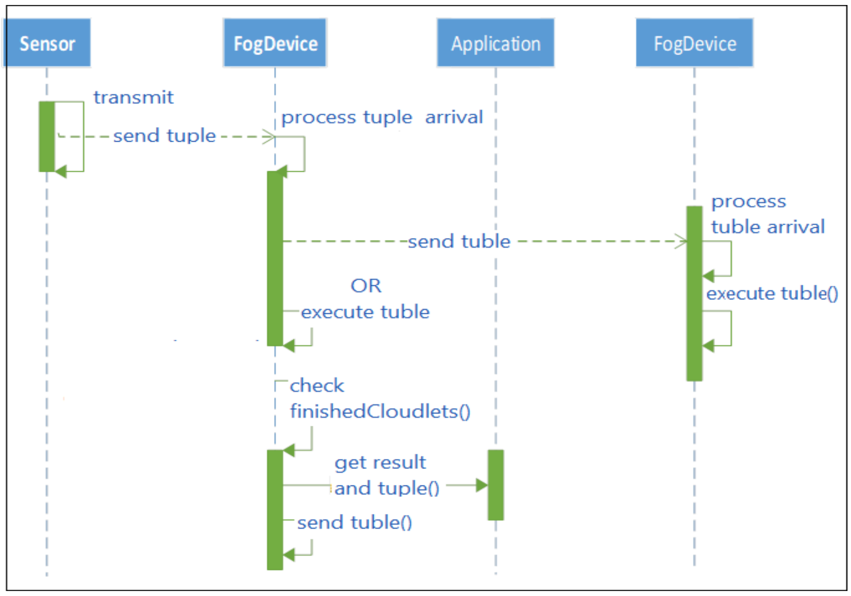

Fig. 3. Sequence diagram of the interaction between the sensor and fog device execution

If the group needs to be forwarded to another fog device, it will be sent immediately without processing. The performance parameters play an essential role in this contribution. These parameters are a set of design criteria that, if changed, would significantly affect the performance, schedule, cost, and risk of the system or facility. The performance parameters targeted by the proposed approach are latency, response time, and network utilization.

Algorithm 1 : optimizing the fog computing performance

```
Start
Initiate *cloudlayer* Initiate *foglayer* Initiate *edgelayer*
Sensing data through the sensor module from the edge layer and add encryption
$m^e = c(moden)$
Applying optimization algorithm
**if** nearest node is available **then**
Decrypt and process
$c^d = (m^e)^d m(moden)$
Send result to an actuator
Send a copy of the result to datacenter
**else if** next nearest node is available **then**
Decrypt and process
$c^d = (m^e)^d m(moden)$
Send result to the actuator
Send a copy of the result to datacenter
**else if** Node is available in maximum waiting time **then**
Decrypt and process
**else**
    Return to cloud Close Resources Results
**end if**
End
```

A. Response Time

The response time is the time it takes for the fog or cloud servers to respond to perform a given task and deliver the results. Increasing the response time value will increase latency and delays, but minimizing the response time will result in better performance in managing and monitoring patient requests over the network. This, in turn, will decrease the response time in fog and cloud, being able to process more requests in a unit of time, improve system performance, and serve patients in less time. The earlier fog servers are migration-based and geographically separated, residing in different locations in the cloud. The leading cloud is far from the fog servers, which handle requests from endpoints and actuators rather than the cloud over the network. In the case study of the patient ECG, the end device has the patient information over the network, which the fog server will pick up. An increase in response time will cause extensive delays. The mathematical equations (1) and (2) are used to determine the value of the response time 47] for fog and cloud, respectively.

$$RT(X_{ij}) = ET(X_{ij}) + Tt(X_{ij}) \qquad (1)$$

Where ET is the execution time, Tt is the transmission time for the request (sending the request with data) and response (receiving the results), X_{ij} refers to the task t_i assigned to the fog node f_j, while X_{icloud} refers to the assignment of the task t_i to the cloud.

$$RT(X_{icloud}) = ET(X_{icloud}) + TrTime(X_{icloud}) \qquad (2)$$

Equations (1) and (2) are implemented using the ”PrintDetails()” method of the Simulator.java class as in APPENDIX A.

B. Network utilization

It is also called network usage and the amount of data targeted by the cloud or fog computing approach. It is one of the most important parameters and a primary factor in dealing with cloud or fog computing architecture.

When the state-of-the-art system is used for patient monitoring and healthcare, it may be insufficient as it is a cloud-based system. In a cloud-based system, the problem of delay is raised. Therefore, it may take some time to convert the generated data to the cloud, and patients need real- time processing for healthcare data in a short time. For this reason, a monitoring system based on the fog computing paradigm is proposed. It processes data quickly, in real-time, and locally near the end devices.

The system resources characterize network utilization regarding data sent and received at the network interfaces. The file ”NetworkUsageMonitor.java” is implemented for calculating the network utilization based on the latency metric in fog and cloud computing, depending on equations (3) and (4). Where N_u denotes the network utilization.

$$N_u(Cloud) = N_u(cloud) + latency \times tupleNwSize \qquad (3)$$

$$N_u(Fog) = N_u(Fog) + latency \times tupleNwSize \qquad (4)$$

While latency is used here to measure the time required for packets to travel from source to destination, tupleNwSize refers to the data packets consumed during the ping process by fog and cloud servers using a tuple of data. As well the network utilization N_u of the execution is also calculated using (3) and (4), using the simulator’s” getNetworkUsage()” method, which is an extension of the NetworkUsageMonitor.java class, as in APPENDIX C.

C. Latency

Latency is the delay of the service once it is requested. It can be due to many factors, such as the architecture of the service was not good; the tasks of fog computing are usually shifted to the edge servers, called cloudlets, and sometimes to the edge devices. The main tasks are split and assigned to specific nodes or hops. In our scenario of an Electronic Health System (EHS)-based application, network latency, and service delays occur once tasks are completed in the cloud instead of in the fog.

The cloud servers meet the requirements at the cloud level, the application processes data late, and the preview of information comes late compared to fog. Where the latency is reduced by adding the cloudlets and edge devices under the central cloud due to the computation of edge and fog. Where the application uses the mathematical model of latency shown in the two equations (5) and (6), for cloud and fog, respectively:

$$L(Cloud) = (pinging + TaskIncreased)/ActiveTime \qquad (5)$$

$$L(Fog) = (pinging + TaskIncreased)/ActiveTime \qquad (6)$$

Where L is the latency and ping is the time it takes for a small data set to be transferred from the patient’s device to a server on the Internet and back to that device. Active time is when the application takes from the first response to receiving the results after processing. The total delay during the execution of the workflow application starts at the time of the request in the fog or cloud environment and is calculated based on the code shown in APPENDIX B.

The cloud often has almost unlimited resources but is far from IoT endpoints, making it unsuitable for providing high-quality services to IoT devices. Edge computing, which has recently emerged as an extension of the cloud, moves massive processing and storage capacity to the network edge, forming an edge layer close to IoT end devices. As a result, many compute-intensive and resource-intensive activities can be moved from resource-constrained end devices to the resource-rich edge layer [48].

D. The Case Study

The case study is a healthcare system in which patients are equipped with ECG sensors. Patients are continuously monitored by the health module located on hospital servers. The nodes continuously process the heartbeat and analyze the patient's health
status. The patients have actuator modules, and the data is transmitted from the actuators to the clusters connected to the module components. This component is connected to the fog implementation server, and the fog is directly connected to the central cloud. The central cloud always receives many patient requests. Still, a problem occurs because the central cloud performs less than the actual fog implementation through the edge server. Moreover, fog computing is centralized computing with decentralized nature of the requests coming from the edge layers. Figure 4 shows how performance is doubly optimized. This work was implemented using a case study of EHS through the fog application network, with the help of fog servers connected to the central cloud. A path is set up for each patient, where each patient has a channel over the Internet. Also, each patient has the following things; a record in the medical history, a previous profile, and all the information needed to make strategic decisions in the future when a doctor is needed. For this purpose, the physician needs live information about the patient on the ventilator and receives live data that the physician module continuously monitors and acts on.

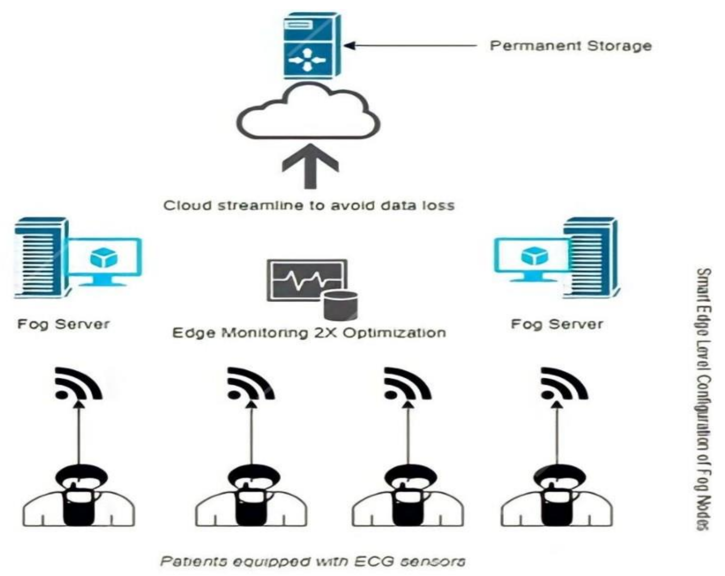
Permanent Storage
Cloud streamline to avoid data loss
Fog Server
Edge Monitoring 2X Optimization
Fog Server
Smart Edge Level Configuration of Fog Nodes
Patients equipped with ECG sensors

Fig. 4. EHS System Optimization

The live data comes from the actuators and the live sensors. The news feeds are also transmitted to the central network of the application. The data is sensitive and the sensors require tracking and identification. For this purpose, the data is transmitted from the fog server to the main application servers in the cloud, but the middle layer is the fog layer with the cloudlets.
The purpose of the cloudlets is to store the additional data and process the tasks required by the fog individually. The nodes in the cloudlet layers' process data much faster than ever before, which reduces many things like downtime and latency.

5. Results and Discussion

This section focuses on the results obtained when applying the proposed methodology. The metrics considered include network utilization, response time, and latency. The values of these metrics were optimized. The results yielded optimal values and the system performance improved clearly. The results are explained and discussed in the following subsections.

A. Response Time

The mathematical implementation of response time is achieved by the node/link bypass model, according to (1) and (2). The results are calculated as in the following two examples. First, by fog:

$RT\ (X_{ij}) = ET\ (X_{ij}) + Tt(X_{ij})$
= 2.0721+8.022=10.09442
Second by the cloud:
$RT\ (X_{icloud}) = ET\ (X_{icloud}) + Tt(X_{icloud})$
= 6.8999+13.79952=20.6995

The response time decreased in the base implementation of fog compared to the cloud. The case study of a patient ECG shows that the nearest fog server picks up the early tasks containing the patient information for ECG recording over the network.

Table 3. Comparison of Cloud and Proposed Fog Model for Response

Task	ET Cloud	ET Fog	Tt Cloud	Tt Fog	Cloud	Proposed Fog
500	6.89998	2.0721	13.79952	8.022	20.6995	10.09442
1000	9.39877	3.5421	15.79443	8.95982	25.19632	12.50192
2000	10.2902	5.5421	20.58064	10.2893	30.87084	15.8314
4000	13.4697	7.04721	26.9394	11.8555	40.4091	18.90272

Table 3 and Figure 5 show the experimental results of cloud and fog concerning the response time when processing a given task. This table shows that the response time increases exponentially as the number of tasks increases. Moreover, comparing the curves for cloud and fog server activities shows a big difference in the response time values of tasks, as in the case of the last task (4000), which represents the total number of server activities to be executed. When all these requests are processed, the cloud requires an average response time of 40.4091, while the time required for the same process by the proposed fog model is only 18.90272. This reduces the time required to process the ECG tasks for patients and triggers through the proposed fog network. In Figure 5, the two curves give a clear picture of the fog and the cloud's response time, clearly comparing their similarities and differences. The ECG trigger task is periodically increased and sent to cloud and fog simultaneously. As mentioned earlier, cloud and fog differ significantly in their response to requests. In addition, Table 3 and Figure 5 show the time spent in the cloud and fog, equal to 20.6995 and 10.09442, respectively. These values indicate that the patient's ECG tasks are processed in fog by a short response time, and the same tasks in the cloud take longer. This gives a clear indicator of response time reduction and gives the optimal time as the number of tasks increases using the fog.

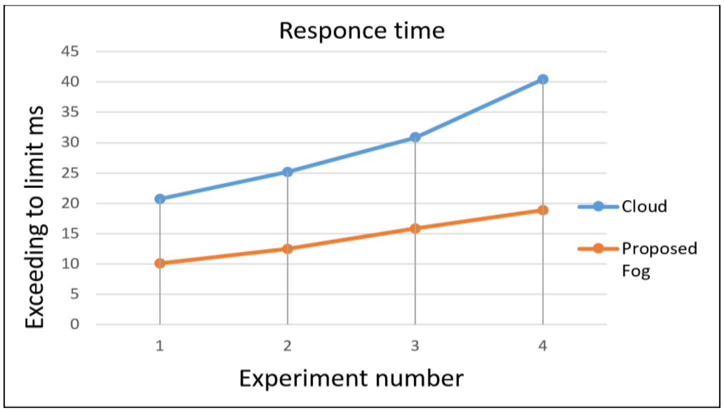

Fig. 5. Response time comparison for cloud and fog computing

B. Network Utilization

Network utilization is the proportion of current traffic to the maximum amount of traffic that can be handled. It reflects how much data the network processes via fog or cloud computing. The average network utilization in the system is the total network consumed by the cloud or fog servers. Figure 6 shows the network utilization values of cloud and fog measured in milliseconds, which shows that the current cloud demand is satisfied in this time frame according to the proposed fog. The values obtained from the proposed fog are 2325.2, which stands for the network used by the fog. The value 75159 represents the network used by the cloud. Table 4 shows the time increases with the number of experiments, and the network utilization in the cloud is much higher than that of the proposed fog. Therefore, it was found that the fog uses much less network than the cloud.

Table 4. Network Utilization

S/No	Task	L-cloud (ms)	L-fog(ms)	tupleNwSize	Nu(Cloud)	Nu(Fog)
1	500	21.474	0.6643	3500	75159	2325.2
2	1000	27.114	8.0973	3500	94901	28340.85
3	2000	36.952	11.343	3500	129332	39701.9
4	4000	48.633	13.464	3500	170216.6	47126.7

The calculation of network utilization, achieved by 4 experiments, the results of Table 4 calculated by (3) and (4) as follows:

- Node experiment 1

$N_u(Fog) = Latency \times tupleNwSize$

$= 0.6643 \times 3500 = 2325.2$

$N_u(Cloud) = Latency \times tupleNwSize$
$= 21.474 \times 3500 = 75159$

• Node experiment 2

$N_u(Fog) = Latency \times tupleNwSize$
$= 8.0973 \times 3500 = 28340.85$

$N_u(Clod) = Latency \times tupleNwSize$
$= 27.114 \times 3500 = 94901$

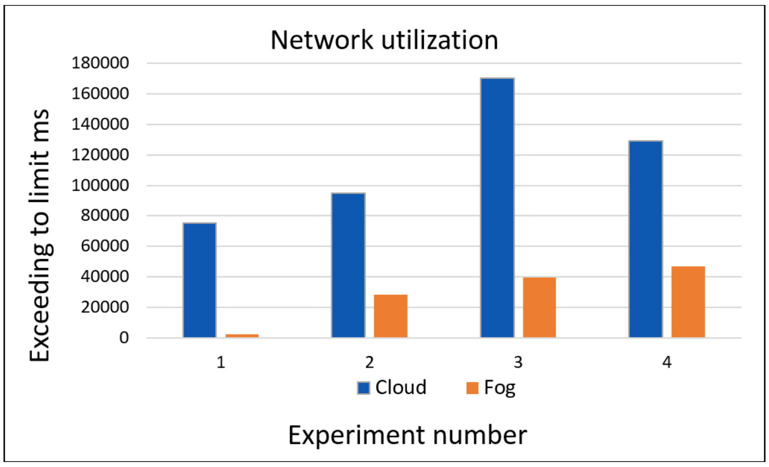

Fig. 6. Comparison between cloud and proposed fog based on network utilization

C. Latency

Latency is the delay in getting the service due to many factors, such as the system architecture and components. Table 5 shows the data and latency results of both cloud and fog. In the table, the active time in the first column indicates when the application responds to display the results. Fog computing tasks are usually outsourced to edge servers called cloudlets, sometimes called edge machines. The main tasks are split and assigned to specific nodes. In the EHS-based application scenario, network latency and service-level delay are reduced when tasks are completed in the cloud instead of in the fog. Cloud servers meet the requirements at the cloud level, the application finishes processing the data late, and the preview of the information comes later than in fog computing, which completes the processing of the data and then displays the information early.

Table 5. Comparison of Latency Results

Active time	**L(Fog)**	**L(cloud)**	**Tasks**	**Fog pinging(sec)**	**Cloud pinging(sec)**
25	22.977	30.836	500	74.430041	270.904
50	21.666	27.064	1000	83.337661	353.2144
100	20.953	24.19	2000	95.345306	419.092
200	20.616	24.732	4000	123.36905	946.5893
400	15.308	17.5	6000	123.43454	1020.345

Mathematical equations (5) and (6) are used to calculate the latency in the application for cloud and fog, respectively. These are two examples for calculating the latency, as follows: First, for cloud:

L(Cloud)=(pinging+task increased)/active time, where L(Cloud) = (270.904 + 500)/25
L(Cloud) = 30.836 seconds
30.8 is the latency delay for the cloud request processed of 500 tasks. Second, for fog: L(Fog)=(pinging+task increased)/active time, where L(Fog)= (74.43 + 500)/25
L(Fog)= 22.977 seconds

In the first experiment, Table 5 shows the results of both cloud and fog when the system has the same number of tasks. The latency is 30.836 seconds for cloud and 22.977 seconds for fog when the number of tasks =500. In this experiment, the fog servers achieved better results, with a difference of 10.2 seconds. In Table 4, the results also indicate that when the number of tasks is 1000, the cloud latency was 27.064 seconds, while the fog latency was reduced to 21.666 seconds. Moreover, the last value gives a clear overview of cloud application latency. With a corresponding increase in the tasks to 6000, the cloud latency was 17.500 seconds, while the fog was reduced to 15.308 seconds when the active time was 400 seconds. As well as with the other experiments. The results of latency are shown in Figure 7. It found that the curve of fog does not increase drastically with the increase of tasks on the cloudlets, and the latency reduces and is controlled appropriately by the fog server.

Based on the previous results and comparisons, it found a clear reduction of the fog's latency compared to the cloud's latency in all values with an increase of tasks and active time, and the processing is much faster with fog than in the cloud. Consequently, the latency was improved and reasonably controlled with the fog server compared to the cloud results.

D. Derived Results for Comparing the Traditional Fog and Modern Fog

We derived results using the simulation software named iFogSim simulation module to optimize performance indicators, which are latency, network utilization, and response time.

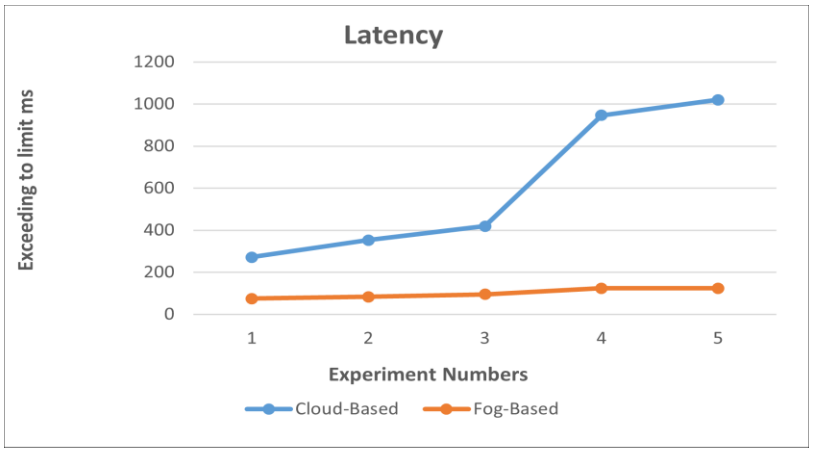

Fig. 7. Comparison between cloud and proposed fog based on latency

Results are quite clear in implementing the whole process over modern Fog and traditional fog with the same structure, let us share some summary of the parameters and discuss the whole application process that obtained in the derived results of the comparison shown in Table 6 and 7.

Table 6. Traditional Fog Results

Response time	Latency	Network utilization
667.65 milliseconds	93.5	0.0303
400.56 milliseconds	96.7	0.0308
500.56 milliseconds	94.5	0.0304
600.23milliseconfds	97.8	0.0308

Table 7. Modern Fog Results

Response time	Latency	Network utilization
500.65 milliseconds	70.5	0.0103
300.56 milliseconds	79.7	0.0180
400.56 milliseconds	80.5	0.0208
490.23milliseconfds	83.8	0.0200

6. Conclusion

The proposed system was implemented in a simulation environment for the EHS using three layers (cloud, fog, and edge). The application was packaged and containerized using fog servers connected to the centralized and compact cloud through the fog application network. The data is transmitted from the fog server to the main application servers in the cloud, but the middle layer is the fog layer, which contains cloudlets. The cloudlets store additional data and perform tasks that need to be done individually by fog. Cloudlets are the sub-layer that complements the main fog layer and allows the cloudlet layer tasks and data node processing to be performed much faster than ever before, reducing many factors like downtime and latency. Optimizing the entire system in the fog environment was achieved. The performance parameters show positive changes in terms of latency, response time, network utilization, and performance of applications using fog computing and over-the-edge. The objective of this paper was to process the main functions of fog in an optimized form so that the optimization and sharing of the processes are done in a fast and accurate method, which was not possible before due to the implementation based only on cloud computing. The results show that the proposed system achieves very significant results. The response time, latency, and network utilization were optimized to a better level. Based on the obtained and discussed simulation results in the previous sections. The proposed system can be applied in the real environment and thus provides excellent results and achieves significant benefits in terms of performance optimization according to the optimization parameters discussed above.

7. Future Work

Another significant modification that can be made in the future in the same E-health system is to increase the number of equivalent nodes. These nodes can process the incoming requests from the operator devices willing to accept the request from the operator. Similarly, the demand coming from the geographically dispersed clinics in the city can be processed by cloudlets since the request originates from there. The proposed simulation system can also be deployed and tested in other domains until it achieves the best possible performance of fog computing.

Furthermore, the performance criteria can be extended to cover other important characteristics. For instance, the wireless network channel can also be another factor that can be considered. In addition, the ability to identify whether the fog node reliability can be used in the node selection criteria, specifically in case the fog node is battery operated.

Funding

This research has been funded by the Scientific Research Deanship at Qassim University, Saudi Arabia, under the number (COC-2022-1-1-J-25647) during the academic year 1444 AH/2022 AD.

Acknowledgment

The author(s) gratefully acknowledge Qassim University, represented by the Deanship of" Scientific Research, on the financial support for this research under the number (COC-2022-1-1-J-25647) during the academic year 1444 AH / 2022 AD".

References

[1] Y. Liu, J. E. Fieldsend, and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," IEEE Access, vol. 5, pp. 25445–25454, Oct. 2017, doi: 10.1109/ACCESS.2017.2766923.

[2] Awotunde, J. B., Bhoi, A. K., & Barsocchi, P. (2021). Hybrid cloud/fog environment for healthcare: An exploratory study, opportunities, challenges, and prospects. In Hybrid Artificial Intelligence and IoT in Healthcare (pp. 1-20). Springer, Singapore.

[3] Tabrizchi, H., Kuchaki Rafsanjani, M. A survey on security challenges in cloud computing: issues, threats, and solutions. J Supercomput 76, 9493–9532 (2020). https://doi.org/10.1007/s11227-020-03213-1

[4] Gurjanov, A. V., Korobeynikov, A. G., Zharinov, I. O., & Zharinov, O. O. (2021). Edge, fog, and cloud computing in the cyber-physical systems networks.

[5] Abdali, T. A. N., Hassan, R., Aman, A. H. M., & Nguyen, Q. N. (2021). Fog Computing Advancement: Concept, Architecture, Applications, Advantages, and Open Issues. IEEE Access, 9, 75961-75980.

[6] Jafari, V., & Rezvani, M. H. (2021). Joint optimization of Energy Consumption and time delay in IoT-fog-cloud computing environments using NSGA-II metaheuristic algorithm. Journal.

[7] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the internet of things: A review", Big Data and Cognitive Computing, vol. 2, no. 10, pp. 1-18, 2018.

[8] M. Chiang, S. Ha, F. Risso, T. Zhang, and I. Chih-Lin, "Clarifying fog computing and networking: 10 questions and answers," IEEE Communications Magazine, vol. 55, no. 4, pp. 18-20, 2017.

[9] S. Chatterjee, "Fog Computing Applications, in Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things," CRC Press, pp 167-186, 2019.

[10] N. A. C and R. Lavanya, "Fog computing and its role in the internet of things, in Advancing Consumer-Centric Fog Computing Architectures," IGI Global, pp. 63-71, 2019. [

[11] B. M. Nguyen, H. T. T. Binh, and B. D. Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," Applied Sciences, vol.9, no.9, p. 1730, 2019.

[12] X. An, J. Su, X, Lu, and F. Lin, ”Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system,” EURASIP Journal on Wireless Communications and Networking, no. 1, pp.1-9, 2018.

[13] S. Svorobej et al., ”Simulating fog and edge computing scenarios: An overview and research challenges,” Future Internet, vol. 11, no. 3, p55, 2019.

[14] E. M. Dogo, A. F. Salami, C. O. Aigbavboa, and T. Nkonyana, ”Taking cloud computing to the extreme edge: A review of mist computing for smart cities and industry 4.0 in Africa,” Edge computing, pp.107-132, 2019.

[15] T. Salman and R. Jain, ”Networking protocols and standards for internet of things,” Internet of Things and Data Analytics Handbook, pp. 215-238, 2017.

[16] P. P. Ray, ”A survey on Internet of Things architectures,” Journal of King Saud University-Computer and Information Sciences, vol. 30, no. 3, pp. 291-319, 2018.

[17] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, ”Fog computing for the internet of things: A survey,” ACM Transactions on Internet Technology (TOIT), vol, 19, no. 2, pp. 1-41, 2019.

[18] Y. Kong, Y. He, and K. Abnoosian, ”Nature-inspired virtual machine placement mechanisms: A systematic review,” Concurrency and Computation: Practice and Experience, vol. 34, no. 11, p. 6900, 2022.

[19] P. Mell and T. Grance, The NIST definition of cloud computing. National Institute of Standards and Technology U.S. Department of Commerce, 2011.

[20] P. Kalagiakos and P. Karampelas, ”Cloud Computing learning,” 2011 5th International Conference on Application of Information and Communication Technologies (AICT), pp. 1-4, 2011.

[21] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, ”A survey on security issues and solutions at different layers of Cloud computing,” The journal of supercomputing, vol. 63, no. 2, pp. 561-592, 2013.

[22] S. S. Islam, M. B. Mollah, M. I. Huq and M. A. Ullah, ”Cloud computing for future generation of computing technology,” 2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), pp. 129-134, 2012.

[23] S. Hashemi and M. Zarei, ”Internet of Things backdoors: resource management issues, security challenges, and detection methods,” Transactions on Emerging Telecommunications Technologies, vol. 32, no. 2, p. e4142, 2021.

[24] S. Hashemi and M. Zarei, ”Internet of Things backdoors: resource management issues, security challenges, and detection methods,” Transactions on Emerging Telecommunications Technologies, vol. 32, no. 2, p. e4142, 2021.

[25] R. P. Padhy, M. R. Patra, and S. C. Satapathy, ”Cloud computing: security issues and research challenges,” International Journal of Computer Science and Information Technology & Security (IJCSITS), vol. 1, no. 2, pp. 136-146, 2011.

[26] S. Yi, C. Li, and Q. Li. ”A survey of fog computing: concepts, applications and issues,” Proceedings of the 2015 Workshop on Mobile Big Data (Mobidata ’15), 2015.

[27] S. Khan, S. Parkinson, and Y. Qin, ”Fog computing security: a review of current applications and security solutions,” Journal of Cloud Computing, vol. 6, no. 1, pp. 1-22, 2017.

[28] R. Cziva, S. Jou ët, D. Stapleton, F. P. Tso and D. P. Pezaros, ”SDN-Based Virtual Machine Management for Cloud Data Centers,” in IEEE Transactions on Network and Service Management, vol. 13, no. 2, pp. 212-225, 2016.

[29] M. Taneja and A. Davy, “ScienceDirect Resource based placement of data analytics algorithm in fog computing,” Procedia - Procedia Computer Science, vol. 00, no. October. pp. 18–20, 2015, [Online] . Available: http://dx.doi.org/10.1016/j.procs.2016.08.295.

[30] S. Yi, Z. Hao, Z. Qin and Q. Li, ”Fog Computing: Platform and Applications,” 2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), pp. 73-78, 2015.

[31] A. Markus and A. Kertesz, ”A survey and taxonomy of simulation environments modelling fog computing,” Simulation Modelling Practice and Theory, vol. 101, p. 102042, 2020.

[32] A. M árkus, P. Gacsi, and A. Kertész. ”Develop or Dissipate Fogs? Evaluating an IoT Application in Fog and Cloud Simulations,” 10th International Conference on Cloud Computing and Services Science, 2020.

[33] R. Mahmud and R. Buyya, ”Modelling and simulation of fog and edge computing environments using iFogSim toolkit,” Fog and edge computing: Principles and paradigms, pp. 1-35, 2019.

[34] E. Marin-Tordera et al., ”What is a fog node a tutorial on current concepts towards a common definition,” arXiv preprint arXiv:1611.09193, 2016.

[35] M. Al-khafajiy, T. Baker, A. Waraich, D. Al-Jumeily and A. Hussain, ”IoT-Fog Optimal Workload via Fog Offloading,” 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion) pp. 359-364, 2018.

[36] H. D. Abdulgalil, ” A Multi-Tier Distributed fog-based Architecture for Early Prediction of pileptic Seizures”, M.S. thesis, University of Waterloo, 2018.

[37] J. Abdelaziz, ”Architectural model for Collaboration in the Internet of Things: a Fog Computing based approach,” M.S. thesis, Universit Montréal, Accueil, 2018.

[38] H. L. M. D. Santos, ”A Multi-tier fog architecture for video on demand streaming,” M.S. thesis, Federal University of Par ´A Institute of Tecnology, 2018.

[39] Z. Hao, E. Novak, S. Yi and Q. Li, ”Challenges and Software Architecture for Fog Computing,” in IEEE Internet Computing, vol. 21, no. 2, pp. 44-53, 2017.

[40] B. Varghese, C. Rea ño and F. Silla, ”Accelerator Virtualization in Fog Computing: Moving from the Cloud to the Edge,” in IEEE Cloud Computing, vol. 5, no. 6, pp. 28-37, 2018.

[41] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, R. Buyya, ”iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments,” Software: Practice and Experience, vol. 47, no. 9, pp. 1275-1296, 2017.

[42] N. Mohan and J. Kangasharju, ”Edge-Fog cloud: A distributed cloud for Internet of Things computations,” 2016 Cloudification of the Internet of Things (CIoT), pp. 1-6, 2016.

[43] S. E. Kafhali and K. Salah, ”Efficient and dynamic scaling of fog nodes for IoT devices,” The Journal of Supercomputing, vol. 73, no. 12, pp. 5261-5284, 2017.

[44] B. M. Nguyen, H. T. T. Binh, and B. D. Son, ”Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment,” Applied Sciences, vol.9, no.9, p. 1730, 2019.

[45] S. Iftikhar, M. M. Ahmad., S, Tuli., D, Chowdhury, M, Xu, S, S, Gill, & S, Uhlig. (2023). HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments. Internet of Things, 21, 100667.

[46] M, Saad, R. I, Qureshi, & A. U, Rehman. (2023, January). Task Scheduling in Fog Computing: Parameters, Simulators and Open Challenges. In 2023 Global Conference on Wireless and Optical Technologies (GCWOT) (pp. 1-6). IEEE.

[47] A. Bonguet and M. Bellaiche, ”A survey of denial-of-service and distributed denial of service attacks and defenses in cloud computing,” Future Internet, vol. 9, no. 3, p. 43, 2017.

[48] G. Mantas, N. Stakhanova, H. Gonzalez, H. H. Jazi, and A. A. Ghorbani, ”Application-layer denial of service attacks: taxonomy and survey,” International Journal of Information and Computer Security, vol. 7, no. 2/3/4, pp. 216-239, 2015.

Appendices

A : Response time source code

```
System.out.println("=========================================");
           System.out.println("Total Response Time:");

     System.out.println("=========================================");

           for (String datafetch :
TimeKeeper.getInstance().getTupleTypeToAverageCpuTime().keySet()){
                System.out.println(datafetch + " <---> 
"+TimeKeeper.getInstance().getTupleTypeToAverageCpuTime().get(datafetc
h));
 }
```

B: Latency source code

```
for (Integer loopId
:TimeKeeper.getInstance().getLoopIdToTupleIds().keySet()) {
System.out.println(getStringForLoopId(loopId) + " ---> "
+ TimeKeeper.getInstance().getLoopIdToCurrentAverage().get(loopId));
if(TimeKeeper.getInstance().getLoopIdToCurrentAverage().get(loopId ) !=
null) {
apploopLatency +=
TimeKeeper.getInstance().getLoopIdToCurrentAverage().get(loopId);
}
   }
```

C: Network utilization source code

```
public class NetworkUsageMonitor {
private static double networkUsage = 0.0;
public static void sendingTuple(double latency,double tupleNwSize) {
     networkUsage += latency*tupleNwSize;
}
public static double getNetworkUsage(){
return networkUsage;
}
  }
```