


RESEARCH

Open Access

Utilization and load balancing in fog servers for health applications



Hasan Ali Khattak¹, Hafsa Arshad¹, Saif ul Islam², Ghufran Ahmed¹, Sohail Jabbar³, Abdullahi Mohamud Sharif^{4*}  and Shehzad Khalid⁵

Abstract

With the evolution of fog computing, processing takes place locally in a virtual platform rather than in a centralized cloud server. Fog computing combined with cloud computing is more efficient as fog computing alone does not serve the purpose. Inefficient resource management and load balancing leads to degradation in quality of service as well as energy losses. Traffic overhead is increased because all the requests are sent to the main server causing delays which cannot be tolerated in healthcare scenarios. To overcome this problem, the authors are consolidating fog computing resources so that requests are handled by foglets and only critical requests are sent to the cloud for processing. Servers are placed locally in each city to handle the nearby requests in order to utilize the resources efficiently along with load balancing among all the servers, which leads to reduced latency and traffic overhead with the improved quality of service.

Keywords: Fog computing, Utilization, Load balancing, Cloud computing

1 Introduction

Cloud computing is an emerging computing technology that uses the internet to maintain large applications and data servers to provide services to clients or end users. It is the delivery of computing services over the internet through computing resources. More precisely, cloud computing is the virtualization and central management of data-centered resources. It refers to remotely managing workloads over the internet in a data center. Cloud computing eliminates the cost of hardware, software, and controls as well as manages the on-site data centers. Data can be mirrored at multiple surplus sites on the cloud provider's network. It generates data backup, provides a disaster recovery, and ensures business continuity with easier and less-expensive services. Cloud providers offer computing services to the clients and charge them based on the usage of these services. Due to inherent problem, some applications cannot work efficiently on the cloud. Problem of low bandwidth arises as data can not be transmitted to the cloud with the frequency as it is being generated. For example, in critical situations like heart

attack when large amounts of data is to be sent back to edge nodes after processing at the cloud, delay cannot be tolerated. Hence, to overcome this issue, the concept of fog computing was introduced.

The term fog computing (also known as fogging), introduced by CISCO, is an extension of cloud computing. It is used to ease wireless data transfer to distributed devices in the IoT network paradigm. Fog computing brings computing resources and application services closer to the edge where data is being generated. The amount of data sent to the cloud for processing and analysis is reduced to a large extent by using fog computing. It reduces the traffic on cloud (i.e., main server) and is able to maintain load between resources, which improves quality of service (QoS) and security.

Fog computing creates a virtual platform providing networking, computation, and storage services close to the level where data is being generated, i.e., in the middle of cloud data centers and end users. These services act as a middle-ware between cloud and fog computing. Since most of the information is now processed locally in a foglet, only summarized information will be transmitted to the cloud and hence bandwidth will be saved to a large extent. It will reduce latency and delays along with packet loss. Although fog computing performs better than cloud

*Correspondence: abdullahi.shariif@uniso.edu.so

⁴Department of Computer Science, University of Somalia, Mogadishu, Makka Al-mukarrama Road, Mogadishu, Somalia

Full list of author information is available at the end of the article

computing, we cannot completely replace cloud with fog; rather, both have to co-exist to support each other whenever required. The general architecture of fog computing is given in Fig. 1, which depicts the working of fog computing. End-user clients are sending requests to the fog layer or middle-ware for communication and storage purposes, where requests are being processed at the fog server and transmitted back to the clients. The requests with greater computing resources will be forwarded to the cloud for processing.

Foglet resource management is an important factor of fog computing. The load among foglets is not balanced (some foglets may be overloaded while others may remain idle). Inefficient resource management and load balancing leads to degradation in QoS and energy management of foglets. FOCAN (Fog Computing Architecture Network) [1] is a multi-tier architecture proposed to overcome these issues. It ensures low energy usage, response time reduction, and reduced network overhead. Abdel et al. in [2] has illustrated the role of IoT in education domain. Along with this, the authors have provided definitions, basic concepts, characteristics, and challenges of IoT. Its role in making efficient and effective decisions is also demonstrated in this work.

This paper integrates fog computing so that:

- Most of the requests are handled by cloudlets and only critical requests are sent to the cloud for further processing.
- Local servers are placed in each city to handle the nearby requests so that resources are utilized efficiently and load is balanced among all the servers.

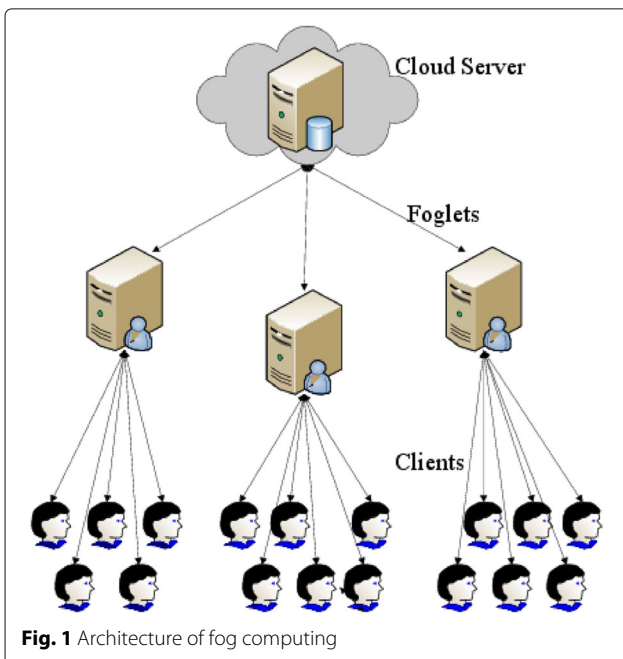


Fig. 1 Architecture of fog computing

- Along with that, latency and traffic overhead can be reduced with the enhanced QoS.

The rest of the document is organized as follows: Section 2 contains brief description of related work. In Section 3, evaluation parameters are defined. Section 4 describes the proposed architecture. Section 5 explains the working of a single foglet. Finally, the document is concluded in Section 6.

2 Literature review

This section describes existing work related to different aspects of fog computing. Fog computing is a widely spread domain and many researchers are exploring it nowadays. We categorize these contributions according to the issues faced in foglet resource management. The authors in [1], have proposed FOCAN (multi-tier structure) for energy consumption, resource utilization, and latency. Low energy usage, response time reduction, and reduced network overhead are achieved while load balancing, security, and privacy are not considered. In [3], the authors proposed definitions for mobile edge computing and motivation of mobile edge computing by considering different applications. This is done to bring cloud services, resources, and mobile edge computing. The authors resolved the issue regarding WAN latency for delay sensitive applications in accessing the cloud resources; however, the problem of traffic overhead is not addressed in this paper.

The problem of load balancing in cloud computing is discussed in [4]. Load balancing algorithms and techniques are used to serve the purpose. Response time is decreased with VM. In [5], a previous allocation state of VM is not saved and the algorithm is executed each time when new request for VM allocation is made. Hence, round robin with server affinity is proposed. With the proposed algorithm, the authors are able to reduce the response time and processing time. The authors in [6] address the problem of response time and processing time. A combination of two algorithms is used: ESCE and throttled algorithm. A comparison of ESCE, throttled algorithm, and round robin is done. From the results, it is clear that ESCE and throttled algorithms performed well, then round robin and time and cost are reduced by using these two algorithms.

The goal of [7] is to locate user demand and to minimize the tremendous growth of mobile traffic. Higher bandwidth cost and energy consumption are considered. The author uses a three-tier mobile fog cloud architecture to achieve the goal. By using this architecture, fog computing absorbs intensive mobile traffic and relieves good data transmission. Data storage, security and privacy issues, and costly and energy-hungry data are the

application areas discussed in [8]. Maintaining and operating sensors directly from cloud servers are non-trivial tasks. Multi-layer telemedicine architecture is proposed using Intel Edison, Raspberry pi, case studies on various types of physiological data and body sensor networks. Reliability is achieved with reduced traffic overhead, and the overall performance is enhanced. Data trimming and resource utilization are the application areas discussed in [9]. The authors presented smart gateway focused communication. A single-hop gateway and multi-hop gateway are used. Efficient resource utilization for better data trimming and pre-processing, and reduced traffic head are achieved. In [10], the authors have proposed the hierarchical model and application architecture, information system infrastructure framework, and resource management mechanism for data backup, data management, and system monitoring. Burden is reduced on network traffic, and system efficiency is also improved while security is not considered.

In paper [11], the authors have considered the security issues faced in fog computing paradigm. The authors have studied a real typical attack (man in the middle attack), analyzing memory consumption and CPU. Data security is more as compared to other SOTA. Real-time scheduling algorithm is proposed in [12], to recover connection failure and to retain services for vehicles that lose the fog server. Control overhead and failure recovery time was decreased by 55%.

The paper [13] deals with three challenges. (1) Design of information sensing nodes in body sensor networks. (2) Collection, storage, and analysis of large amounts of heterogeneous data. (3) Energy efficiency of edge devices. The authors propose a service-oriented fog computing architecture to overcome these challenges of data reduction, low power consumption, and high efficiency. Dubey et al. built a prototype using Intel's Edison in order to show the efficiency of the proposed architecture. Achievements of the authors include reduction in storage and power along with the reduced logistics requirements. The main focus of the authors in [14] is to enhance the IoT-based health monitoring systems used for diversified environments. The authors proposed an IoT-enabled healthcare system architecture to demonstrate the efficiency of bandwidth utilization, emergency notification, and quality of service assurance. To determine the efficiency of fog computing in healthcare applications, it is implemented on a case study of ECG. This paper implemented different fog computing services like location awareness, interoperability, graphical user interface with access management, distributed database, and real-time notification mechanism.

The authors in [15] are dealing with the problem of resource management. A methodology for management and estimation of resources is proposed which is known as

relinquish probability. How many resources are going to be used and whether all the requested resources are consumed or not, this cannot be predicted. With the model proposed by Aazam et al., one can determine the right amount of resources required which results in reduction of resource wastage and profit-cut for CPS and fog. Authors in [16] proposed the new fall detection algorithm as early algorithms for fall detection have too much false alarm rate and missing rate. However, with the proposed algorithm, high specificity and high sensitivity is achieved with a minimum response time and energy consumption. The paper [17] presents the fog computing architecture for emergency alerts. A lot of work has already been done on emergency management despite that the architecture proposed in this paper is simple and efficient. By a single button click, users can send alert and then the application decides on its own which department should be informed [18]. Moreover, it automatically informs the patients family by sending them a message. Overall delay with fog computing is reduced six times than other cloud cases. In paper [19], iFogSim is introduced, modelled, and simulated in IoT, edge, and fog environment. In particular, the authors described two case studies and demonstrated effectiveness of iFogSim.

A framework is proposed in [20] for supply chain management on the basis of IoT technologies, in order to deal with the incomplete information effectively. In [21], a whale optimization algorithm along with local search strategy is proposed for dealing with the scheduling problems. A new variant of whale optimization algorithm is proposed in [22]. Experiments are performed to prove the proficiency of the proposed algorithm. The authors in [23] have presented the selection criteria for suppliers. Quality is proved to be the most important criteria in the selection of the suppliers. To deal effectively with vague, imprecise, and inconsistent information, the neutrosophic set concept is applied to the linear programming problems [24]. Two-level clustering algorithm is proposed in [25] for the detection and localization of duplicated areas in a digital image. The study revealed in [26] have provided solutions for quadratic assignment problems. Three-way decisions on neutrosophic sets are applied in [27]. The effective and efficient rule of the neutrosophic set for supplier selection is also illustrated in this work.

The state of the art work is given in Table 1 which clarifies that only Aazam et al. in [9] and [4] are dealing with load balancing. Likewise, other parameters are also considered by many researchers. The aim of this paper is to efficiently utilize the resources so that cost of utility can be reduced along with the improved QoS. Section 3 contains the description of the parameters which are discussed in Table 1.

3 Evaluation parameters

This section gives a brief description of the different evaluation parameters along with their importance in the healthcare domain.

3.1 Load balancing

It refers to the distribution of application or network traffic among different servers in order to enhance the capacity and reliability of the applications. It is the distribution of the task performed by a single computer into multiple computers so that more work gets done at the same time. By this distribution of workloads and the computing resources, we can manage the workload demands in a better way by allocating resources (requests in this case) among multiple servers and will serve users rapidly [28]. This will lead to a high availability and increased performance rate. The aim of this research is to efficiently utilize load which can be done by distributing workload among different servers. If one server is too busy in handling requests from the clients and other servers are idle or have a less amount of requests, then it will transfer some of the load on the nearby server which has no or less requests. Hence, by this distribution of workload, we can achieve efficient utilization of energy and resource consumption.

3.2 Latency/response time

Latency refers to delays which usually occur when any component of the system waits for another component to complete the task. Basically, it is the time taken by the processor to handle the request, i.e., from the moment of

transmission till the time it is received back to the client after all the processing [29]. In case of any delay in this processing time, it is considered as latency. It is among one of the drawbacks of the cloud computing which is handled by fog computing. In the scenario discussed in this paper, delays can not be tolerated as in life-threatening situations delays may cause any mishap.

3.3 Quality of service

It is the ability to provide better services to network traffic over different technologies. The goal of QoS is to allocate the lead containing dedicated bandwidth, managed and controlled latency, and jitter. QoS is not a one time deployment in a varying network environment, rather it is an ongoing and fundamental part of a network design.

3.4 Bandwidth

It is the volume of information that can be transmitted per unit of time. The maximum amount of data that can be transmitted over a specific network or internet in a given amount of time that the transmission medium can handle is termed as bandwidth. It only describes the speed of the network and does not tell how fast data is moving from one location to another. The amount of bandwidth required depends on what you are planning to do with your internet connection. The higher the bandwidth, more data are transmitted, and hence, in a particular time, more processing of data can be done due to the large amount of data transmission. One can also limit the bandwidth for a certain task. This control of bandwidth is set

Table 1 State-of-the-Art Work to compare contributions by different aspects

Reference papers	Traffic overhead	Quality of service	Loadbalancing	Latency/response Time	Energyconsumption	Bandwidth
[1]	✓	✓	×	✓	✓	✓
[3]	×	✓	×	✓	×	✓
[4]	×	×	✓	✓	×	×
[5]	×	×	×	✓	×	×
[6]	×	×	×	✓	×	×
[7]	✓	×	×	×	✓	✓
[8]	×	×	×	✓	×	✓
[9]	✓	×	✓	×	×	×
[10]	✓	✓	×	×	×	×
[11]	×	×	×	×	×	×
[12]	×	×	×	✓	×	✓
[13]	×	×	×	×	✓	✓
[14]	×	✓	×	✓	×	✓
[15]	×	×	×	✓	×	×
[16]	×	×	×	✓	✓	×
[17]	×	×	×	✓	×	×
[33]	×	×	×	✓	×	✓

Table 1 State-of-the-Art Work to compare contributions by different aspects (Continued)

Parameters							
Security	Privacy	Support of mobility	Interoperability	Data storage	Network management	Resource management	Jitter
x	x	✓	x	✓	x	x	✓
x	✓	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
✓	✓	x	✓	✓	x	x	x
x	x	x	x	x	✓	✓	x
✓	x	x	x	x	x	✓	x
✓	✓	x	x	x	x	x	x
✓	✓	x	x	x	✓	x	x
x	x	x	x	✓	x	x	x
x	x	x	✓	x	x	x	x
x	x	x	x	x	x	✓	x
x	x	x	x	x	x	x	x
x	x	x	x	✓	x	x	x
x	x	x	x	✓	x	x	x

by an internet service provider to limit particular traffic during a certain period of time in order to reduce traffic congestion.

3.5 Traffic overhead

It refers to the amount of extra resources that do not have any direct relationship with the production. The amount of processing time required by the system including the operating system, the utility which supports the application programs, and the installation of any of the particular feature will add to the proportion already needed by the program. It is also defined by Martin et al. in [30] as the processing time of a processor in which it is engaged in the transmission or reception of each message; during this period of time, the dedicated processor cannot perform any other operations.

3.6 Energy consumption

This is the amount of energy used in a particular process or system, or energy consumed by an organization or a society. It is the total amount of energy required to provide the services to the end users. Energy conservation will lead to more efficient systems thus reducing the wastage of resources.

3.7 Security

It refers to securing the data, most probably sensitive data, of the end users or clients of the system. The protection of the data to make sure that only authorized personnel

have access over the data or services provided by a certain system and to ensure the safety against attacks. It is one of the biggest challenges which fog computing is facing, and in order to overcome this access, control should be applied.

3.8 Privacy

Utilization of data while protecting the privacy of the individuals. It refers to the anonymity of an individual and determines whether the data or information of any individual or organization can be accessed by third parties or not, and when, how, and to whom this information is to be revealed.

3.9 Support of mobility

To facilitate traffic forwarding from one node to another which results in change of location of the information. It is useful to handle and allocate resources efficiently among fog nodes [31]. In this work, it is referred as the ability to transfer some load from an over burdened fog node to the idle one. The mobility has an important influence on communication as well [1].

3.10 Interoperability

It refers to the ability of system or software to use the information even after exchange of the data or information. This use of the same tool or software on a variety of platforms is considered as interoperability. In this scenario, if we move or exchange data from one foglet to

another, then it would be usable and useful for transmitting the same information.

3.11 Data storage

Recording of information in a storage medium for future use is termed as data storage. With the increasing demand, data storage and data processing in the IoT have become an issue. To resolve this problem, utilization of cloud computing was introduced which was later replaced by fog computing.

3.12 Network management

It refers to monitoring and managing a wide range of computer networks which might be a burden for fog computing unless some of the techniques are applied on it. Applying these techniques on fog computing may be a challenging task and may lead to mismatch with the goals of efficiency and latency.

3.13 Resource management

It is the efficient and effective management of all the available resources in a best possible way. It is responsible for allocation of resources and maintenance of resource pool in a distributed fashion.

3.14 Jitter

It is the delay between the received packets. It is considered as the variation in the data flow between two systems which might occur because of network congestion. Jitter can be reduced with fog computing.

4 Proposed architecture

The architecture proposed in this paper contains two servers, namely, cloud server and fog server. Patients will send the request to the fog server via IoT. The IoT is the network of objects and the connectivity among these objects which allow them to connect and exchange data. Figure 2 depicts that the requests from patients of different cities are handled by different servers. All the requests from the cluster of Quetta are received by foglet A, foglet B is handling the requests from Karachi cluster, similarly foglet C is dealing with the requests coming from the cluster of Islamabad, and requests from Lahore cluster are handled by foglet D. In return of the request received, an acknowledgement will be sent to the patients that the request is received. On the same fog layer, status of the patient will be checked and if the condition of patient is critical and after making the copy of that request, it will be forwarded to the cloud server without any delay. For example, the heart ECG of the patient is determined if the condition of the patient is critical, then it will be sent to the cloud for further processing; otherwise, the fog server will handle this request. If any of the server is overloaded, then the requests will be shifted to another nearby server

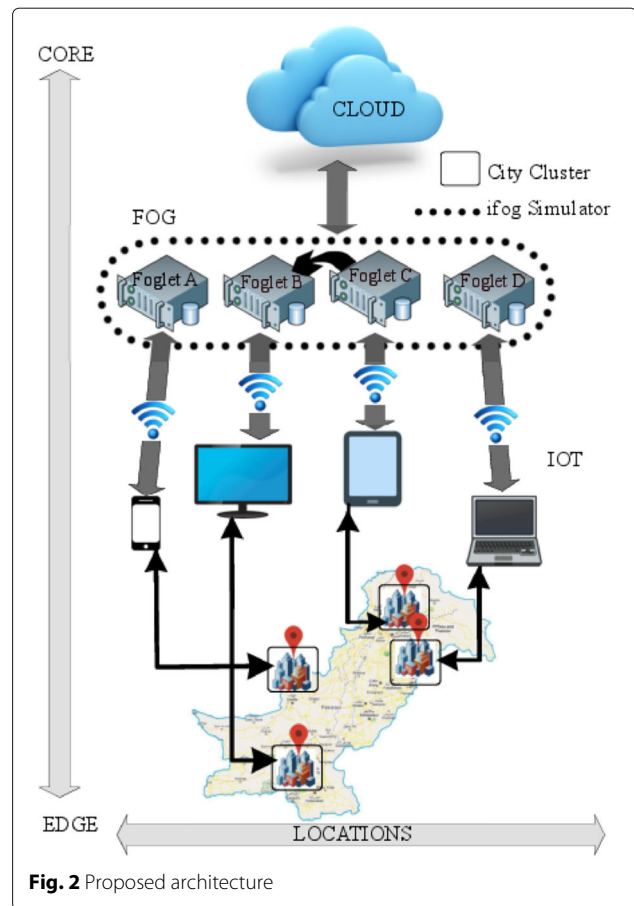


Fig. 2 Proposed architecture

which will respond to the request according to the condition of the patient. Under normal circumstances, the request will be handled by the fog servers; however, in case of a critical condition of the patient, the request will be forwarded to the cloud server for processing. Fog layer which is the middle-ware is maintaining the copy of the request for future references and in order to maintain the record of that patient. Cloud server will generate alerts, and according to the condition of the patient, it will send some precautionary measures to the patient. Fog layer is also handling the load balancing. Capacity of each foglet is determined according to the capacity of each server requests handled. If the number of requests on a server exceeds the level, then it will be forwarded to some other nearby server and hence balancing of incoming requests will be done.

5 Methods/experimental setup for foglets

In this section of the paper, step-wise flow of activities is given. Working of a single component or a foglet is described in detail in Fig. 3.

For experimental validation, we have used the iFogSim framework, which is an industry standard for the development of fog models along with IoT integration. Moreover,

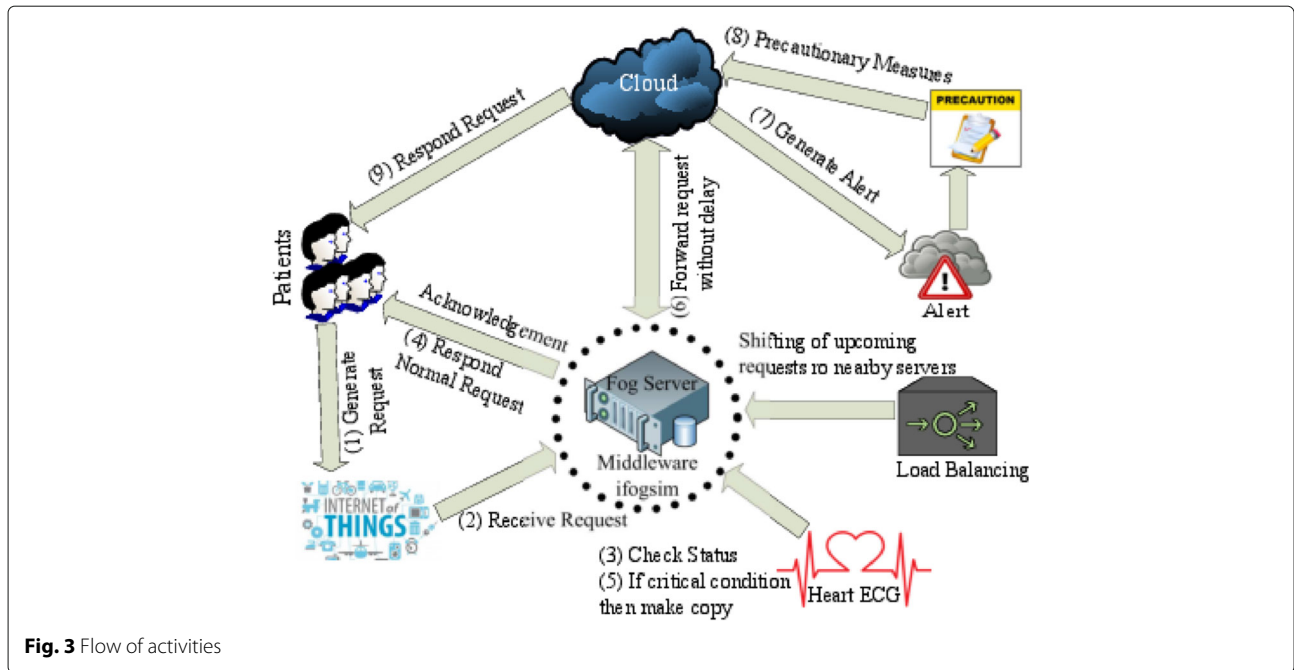


Fig. 3 Flow of activities

it provides enhanced capabilities to measure the impact of resource management techniques through important parameters such as latency, energy consumption, and cost (<https://github.com/harshitgupta1337/fogsim>). In particular, authors have described two useful case studies and demonstrated the effectiveness of iFogSim. There are a number of future directions that can enhance iFogSim and resource management techniques in the context of IoT.

5.1 Generate request

In the first step of the proposed architecture, patients are going to send the requests to the fog server through IoTs which include objects connected to the internet. Both normal and critical patients can generate request and send that to the server for processing.

5.2 Receive request

The fog server will receive the request from the patient and an acknowledgement will be sent from the fog server to the patient that the request is received and it is being processed.

5.3 Check status

The status of the patient will be checked in this step. If the condition of the patient is normal, then the fog server will handle this request in order to avoid the traffic on the cloud server. Otherwise, the request will be sent to the cloud.

5.3.1 Respond normal request

After checking the status of the request, its response will be sent back to the patient. The fog server will respond

the request of the normal patient. A normal heart rate of an adult and aged people ranges from 60 to 100 beats per minute. However, the normal heart rate for children is between 70 and 100 beats per minute. These ranges may vary according to the health condition of each person. Hence, all the requests which lie between these ranges are considered normal and these requests will be handled by the fog server.

5.3.2 Critical patient

If the condition of the patient is critical, then the fog server will make a copy of the request received from patient and after that it will forward the request to the cloud without any delay.

5.4 Response time

The request from the patient who is in critical condition is forwarded to the cloud server without any delay because in safety critical systems or in life-threatening conditions delays are not affordable. A request is sent to the cloud server because most of the requests are handled by the fog server so that there would be less traffic on the cloud. Hence, within the minimum possible time, a request can be sent to the cloud to avoid any mishap.

5.5 Alert

After receiving the request, the system will generate an alert considering the condition of the critical patients.

5.6 Precautions

Precautionary measures regarding the condition of the critical patient are then generated and sent to the patient

that he needs to follow immediately. For instance, if the heart beat of a person exceeds the normal range, then the person will be notified through an alert and precautionary measures will be suggested to them which are expected to be followed by the patient.

5.7 Respond request

Request could be responded for both stable and unstable patients.

5.7.1 Stable condition

After precautionary measures, if the patient gets stable, then they will be informed that their condition is normal now and that they can continue with their normal routine.

5.7.2 Unstable condition

In case the condition of the patient does not get stable, then they will be suggested to rush towards the hospital immediately in order to avoid any mishap.

6 Experimental analysis

Computation of heart condition according to patient category is calculated through an algorithm for computing heart condition according to patient category in Algorithm 1. Heart rate (Hr) and patient categories (R1, R2) are given as an input. On the other hand, result is stored in Res. In line 1 of the algorithm, a function named Fog-computation is created in which three parameters are passed. R1 is the category of aged people which ranges from 60 to 100 while R2 is the category of children which ranges from 70 to 100. If the heart rate of the patient belongs to the first category R1, as shown in line 2, then it will check whether it lies between the normal range of the aged people. If so, then it will respond the request, or else it will consider it as a critical request. If the heart rate belongs to the second category of patients, then line 8 to line 14 will be executed in which it will check the normal heart beat range of children and it will then handle the request accordingly. After that, it will return the result which is stored in Res and function will end in line 16.

Algorithm 1: Algorithm for Computing Heart Condition according to Patient Category

```

Input:  $R_1, R_2, H_r$            ▷ Heart Rate and Patient Categories.
Output: Res                  ▷ Calculated Result
1 Function Fog-computation ( $R_1, R_2, H_r$ ):
2   if  $H_r \in R_1$  then
3     if  $H_r \geq 60 \ \& \ H_r \leq 100$  then
4       Res  $\leftarrow$  Normal;
5     else
6       Res  $\leftarrow$  Critical;
7     end
8   else
9     if  $H_r \geq 70 \ \& \ H_r \leq 100$  then
10      Res  $\leftarrow$  Normal;
11    else
12      Res  $\leftarrow$  Critical;
13    end
14  end
15  return Res
16 End Function

```

Similarly, algorithm for utilization of fog servers shown in Algorithm 2 is for the utilization of foglets in which request (Req) and foglets (Fog) are taken as an input and list of responses (Res) is considered as an output. A function namely Utilization is created in which one parameter Req is passed. Since all the foglets belong to the fog server, hence, a condition is applied that if the foglet is not overloaded then its response will be saved and matched with the patients category and condition. If the response is normal, then it will return the results, or else the request will be forwarded to the cloud. However, if the foglet is overloaded, then the request will be forwarded to another foglet or fog node where the request is processed and it will be responded according to the condition of the patient. Hence, load is balanced among multiple servers while utilizing the resources efficiently.

Algorithm 2: Algorithm for Utilization of Fog Servers

```

Input: Req, Fog              ▷ List of Requests and Fog servers
Output: Res                  ▷ List of Response
                               ▷ Req contains Heart rate and patient category
1 Function Utilization (Req):
2   foreach  $fog_i \in Fog$  do
3     if  $fog_i$  is not overloaded then
4       foreach  $Res_i \in Res$  do
5          $res_i \leftarrow fog_i(req_i)$  ▷ Calling function to request fog
6         node
7         if  $res_i == Normal$  then
8           return  $res_i$ 
9         else
10         $res_i \leftarrow cloud(req_i)$  ▷ Calling function to request
11        Cloud
12      end
13    end
14  else
15    if  $fog_i$  is overloaded then
16      foreach  $Res_j \in Res$  do
17         $fog_j \leftarrow fog_i(req_i)$ 
18         $res_j \leftarrow fog_j(req_i)$  ▷ Calling function to request
19        another fog node
20        if  $res_j == Normal$  then
21          return  $res_j$ 
22        else
23           $res_j \leftarrow cloud(req_i)$  ▷ Calling function to
24          request Cloud
25        end
26      end
27    end
28  end
29  return Res
30 End Function

```

7 Results and discussion

The evaluations are performed using a C#-based Edge-of-Things (EoT) simulation environment that provides the IoT devices, networking infrastructure, edge/fog nodes, and cloud data centers. The simulation setup is inspired from CloudSim [32] and IFogSim [19] simulation environments and provides all the basic and advanced features for simulating an EoT environment. We have considered various types of bio-medical sensor jobs (glucose monitoring, cochlear implant, blood saturation, movement sensor, ECG, and EMG). Fifteen thousand jobs are considered for simulation with 10 heterogeneous fog computing devices.

7.1 Internal processing time

It refers to the time taken by a bio-sensor job on a fog resource. It shows the time a job occupies a fog resource. Figure 4 shows the internal processing time of different bio-sensor data when served by the fog server. The *x*-axis represents the bio-sensor types and *y*-axis shows their corresponding computational time. The internal processing time is evaluated in milliseconds. It is exhibited that the movement sensor data takes more internal processing time compared to other bio-sensor data in the given scenario. The ECG and cochlear implant sensors have the lower processing time. The computational time depends on different factors such as data size and computational complexity of the job. Moreover, fog servers are heterogeneous with varying processing capabilities. Consequently, the computational power of fog servers also have a significant impact on internal processing time of jobs. Similarly, the load of jobs on a fog server can effect its performance and ultimately the internal processing time. It should also be noted that we are considering a fog-cloud environment where the heavy jobs are entertained by the cloud. Here, we have only evaluated the internal processing time of jobs on fog infrastructure.

7.2 Power consumption

Power consumption is considered one of the major constraints in large-scale distributed systems. The usage of fog computing in the medical sector is one of the best proposed solutions for computational tasks due to the delay sensitive nature of medical applications. Therefore, it is critical to analyze the impact of medical jobs on the power consumption of fog resources. Power consumption mainly depends on utilization. Figure 5 shows the power consumed by different types of bio-sensor data when served at the fog resources. The *x*-axis depicts the type of bio-sensors that generate the jobs to be processed, whereas the *y*-axis represents their corresponding power

consumption in watts. It shows how different types of jobs consume fog resources. It is represented that EEG and cochlear implant sensor data exhibits higher power consumption in the underlined environment. The jobs that produce higher usage of servers consume more energy on fog resources. Figure 6 presents the power consumed by a pool of fog servers deployed in the proximity of biomedical data-generating sources. It exhibits the trend of power consumed at the fog layer for different types of medical sensor data. When a fog server is idle, it consumes a constant amount of power and rest of the power is proportional to its utilization. Here, the power consumption of fog servers is considered; however, the jobs that have higher computational requirements are forwarded to the cloud. On the fog layer, there is a cooperation among servers to serve a particular job. This activity results in utilization of additional resources. Consequently, it causes additional power consumption.

7.3 Response time, propagation time, and internal processing time

Response time is the total time of all type of delays involved in serving of bio-medical request. It includes the propagation delay and processing delay. Figure 7 shows the evaluation of performance metrics of underlined IoT-fog-cloud environment. These metrics are considered most critical specially in ambient assisted living (AAL) environment. Sophisticated services are required in pervasive healthcare. The abovementioned metrics are evaluated in milliseconds. It shows that propagation time and processing time majorly contributes in the total time of serving bio-sensor jobs. However, internal processing time is playing a major role in average response time compared to propagation time, in the given scenario. This is due to the forwarding of jobs towards the nearest available fog server. Consequently, it experiences lower propagation delay. The propagation time mainly depends on the

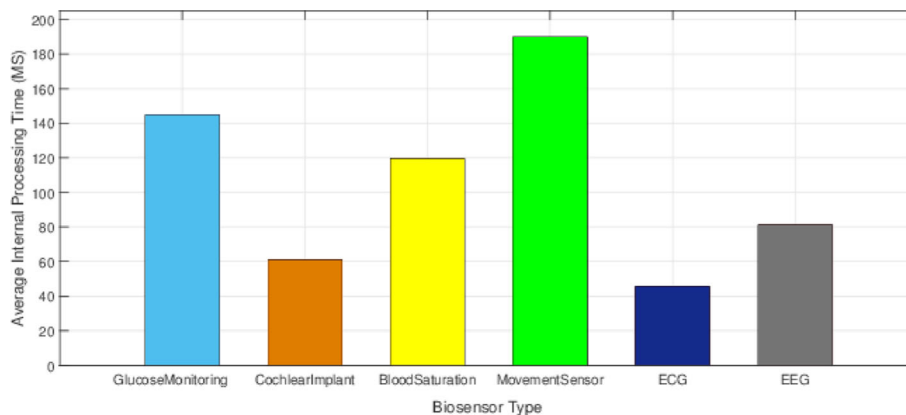


Fig. 4 Average internal processing time

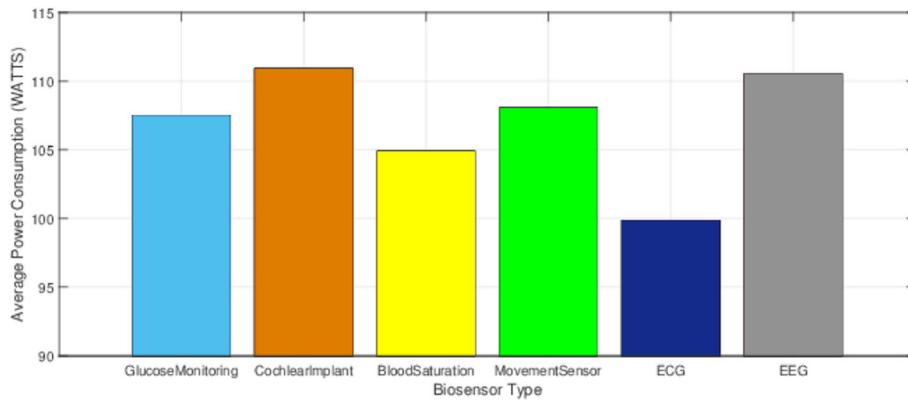


Fig. 5 Power consumption of various bio-sensors

available bandwidth, data rate, and network conditions. The congestion on network nodes results in higher propagation time. Similarly, the high frequency of traffic also has an impact on transmission of data from the generating source to the serving destination. As discussed earlier, the internal processing time depends on computational power of fog resource and resource requirements of job. In a performance point of view, lower values of all abovementioned metrics are preferred for better service provisioning. Higher delays in a bio-medical scenario may degrade the overall performance and can result into health hazards.

8 Conclusion

Fog computing being the extension of cloud computing provides ease in wireless data transfer to the distributed devices in the IoT paradigm. It serves as a middle-ware by bringing computing resources and application services closer to the edge where data is being generated. Previously, with the cloud computing, all the data was being

sent to the cloud for processing which results in delays and latency was not considered. However, with the fog computing, only a limited amount of information will be transmitted to the cloud; hence, bandwidth will be enhanced to a large extent with the reduced latency and delay along with packet loss. The aim of this research is efficient utilization of resources. By placing fog servers between cloud and end users, delays can be reduced. As in healthcare scenarios, delays cannot be tolerated. We are distributing the load evenly on all the servers so that the overburdened servers may not lead to breakdown and all servers are having approximately equal amounts of load. This is done by shifting the load from the overburdened server to the nearby server which is idle or which has less load. Even distribution of load on the fog layer will lead to load balancing and efficient utilization of resources. In this work, we examined how utility is affected by various parameters. More specifically, we have examined fog server utilization. In order to evaluate utilization of fog nodes, we contribute to the capabilities of the iFogSim tool

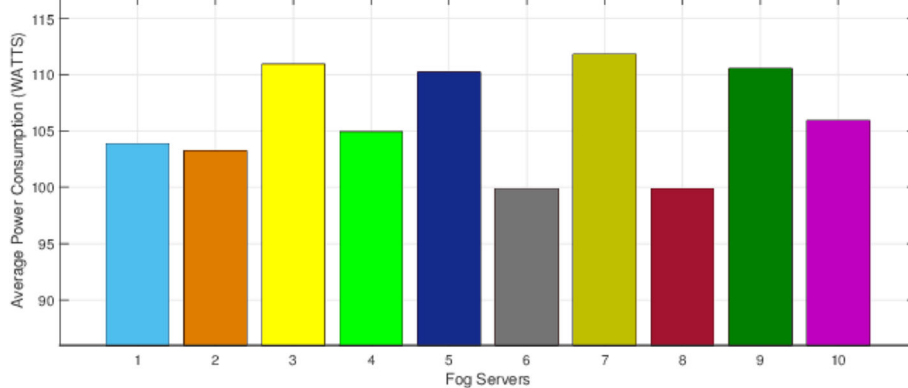


Fig. 6 Average power consumption of fog nodes

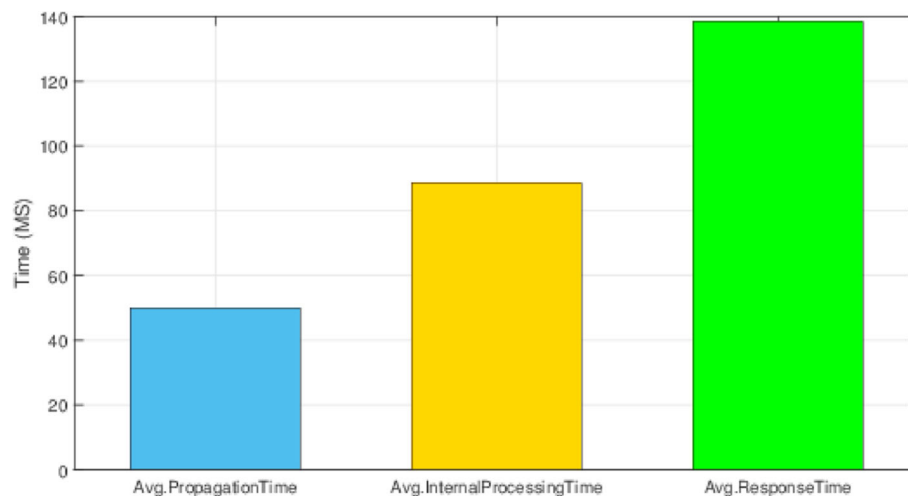


Fig. 7 Power consumption of sensor nodes

over the fog layer by balancing the load among the fog nodes. We are considering load balancing and latency in this work. Likewise, other parameters discussed in Table 1 can also be explored in future for this type of research work.

Abbreviations

AAL: Assisted ambient living; ECG: Electrocardiogram; EEG: Electroencephalography; EMG: Electromyography; ESCE: Equally spread current execution algorithm; FOCAN: Fog computing architecture network; IoT: Internet of things; QoS: Quality of service; SOTA: State-of-the-art techniques; VM: Virtual machine; WAN: Wide area network

Acknowledgements

COMSATS University Islamabad provided support for conducting the experiments.

Funding

This work is funded under a HEC funded project Start up Research Grant Project no 1674.

Availability of data and materials

Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Authors' contributions

HAK conceived the main idea and SI designed the experiments. HA and SI performed the analysis with constructive discussions with GA. HAK and SI provided mathematical guidance for this paper. SJ, AMS, and SK contributed to the structuring and reviewing of the manuscript. All authors have read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Author details

¹Department of Computer Science, COMSATS University Islamabad, Park Road Tarlai Kalan, Islamabad, 44500, Islamabad, Pakistan. ²Department of Computer Science, Dr. A. Q. Khan Institute of Computer Sciences and Information Technology, Rawalpindi, Sumbal Gah Kahuta, 47320 Rawalpindi, Pakistan.

³Department of Computer Science, National Textile University, Faisalabad, Sheikhupura Road, Manawal, 37610 Faisalabad, Pakistan. ⁴Department of Computer Science, University of Somalia, Mogadishu, Makka Al-mukarrama Road, Mogadishu, Somalia. ⁵Department of Computer Science, Bahria University Islamabad, Shangrilla Rd, E-8/1, 44500 Islamabad, Pakistan.

Received: 11 July 2018 Accepted: 5 March 2019

Published online: 08 April 2019

References

1. P. Naranjo, G. Vinueza, Z. Pooranian, M. Shojafar, M. Conti, R. Buyya, FOCAN: A Fog-supported Smart City Network Architecture for Management of Applications in the Internet of Everything Environments. arXiv preprint arXiv: **1710**, 01801 (2017)
2. M. Abdel-Basset, M. Mohamed, G. Manogaran, E. Rushdy, Internet of things in smart education environment: Supportive framework in the decision-making process. *Concurrency Comput. Pract. Experience*, e4515 (2017)
3. E. Ahmed, M. H. Rehmani, Mobile edge computing: opportunities, solutions, and challenges. *Futur. Gener. Comput. Syst.* **70**, 59–63 (2017)
4. R. Soumya, A. D. Sarkar, Execution analysis of load balancing algorithms in cloud computing environment. *Int. J. Cloud Comput. Serv. Archit. (IJCCSA)*. **2**(5), 1–13 (2012)
5. M. Komal, A. Makroo, D. Dahiya, Round robin with server affinity: a VM load balancing algorithm for cloud based infrastructure. *J. Inf. Process. Syst.* **9**(3), 379–394 (2013)
6. N. Shaveta, G. Raj, Comparative analysis of load balancing algorithms in cloud computing. *Int. J. Adv. Res. Comput. Eng. Technol.* **1**(3), 120–124 (2012)
7. T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, L. Sun, Fog computing: Focusing on mobile users at the edge. arXiv preprint arXiv: **1502**, 01815 (2015)
8. D. Harishchandra, A. Monteiro, N. Constant, M. Abtahi, D. Borthakur, L. Mahler, Y. Sun, Q. Yang, U. Akbar, K. Mankodiya, in *Handbook of Large-Scale Distributed Computing in Smart Healthcare*. Fog Computing in Medical Internet-of-Things: Architecture, Implementation, and Applications (Springer, Cham, 2017), pp. 281–321
9. M. Azam, E. Huh, in *2014 International Conference on Future Internet of Things and Cloud, Barcelona*. Fog Computing and Smart Gateway Based Communication for Cloud of Things, (2014), pp. 464–470. <https://doi.org/10.1109/FiCloud.2014.83>
10. H. Xingye, L. Xinming, L. Yinpeng, in *2010 International Conference on Computational and Information Sciences, Chengdu*. Research on Resource Management for Cloud Computing Based Information System, (2010), pp. 491–494. <https://doi.org/10.1109/ICCIS.2010.127>
11. I. Stojmenovic, S. Wen, in *2014 Federated Conference on Computer Science and Information Systems, Warsaw*. The Fog computing paradigm:

- Scenarios and security issues, (2014), pp. 1–8. <https://doi.org/10.15439/2014F503>
12. S. Park, Y. Yoo, Network Intelligence Based on Network State Information for Connected Vehicles Utilizing Fog Computing. *Mob. Inf. Syst.* **2017**, 9 (2017). <https://doi.org/10.1155/2017/7479267>
 13. S. Yi, Z. Hao, Z. Qin, Q. Li, in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Washington, DC. Fog Computing: Platform and Applications, (2015), pp. 73–78. <https://doi.org/10.1109/HotWeb.2015.22>
 14. H. Dubey, J. Yang, N. Constant, A. M. Amiri, Q. Yang, K. Makodiya, in *Proceedings of the ASE BigData & SocialInformatics 2015 Article No. 14 Kaohsiung, Taiwan - October 07 - 09, 2015*. Fog data: Enhancing telehealth big data through fog computing, (2015)
 15. T. N. Gia, M. Jiang, A. Rahmani, T. Westerlund, P. Liljeberg, H. Tenhunen, in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, Liverpool*. Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction, (2015), pp. 356–363. <https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.51>
 16. M. Aazam, E. Huh, in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, St. Louis, MO. Dynamic resource provisioning through Fog micro datacenter, (2015), pp. 105–110. <https://doi.org/10.1109/PERCOMW.2015.7134002>
 17. C. Yu, S. Chen, P. Hou, D. Brown, in *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on*. FAST: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation (IEEE, 2015), pp. 2–11
 18. S. Yingjuan, G. Ding, H. Wang, H. E. Roman, S. Lu, in *Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech), 2015 2nd International Symposium on*. The fog computing service for healthcare (IEEE, 2015), pp. 1–5
 19. G. Harshit, 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. *Softw. Pract. Experience.* **47**(9), 1275–1296 (2017)
 20. M. Abdel-Basset, G. Manogaran, M. Mohamed, Internet of Things (IoT) and its impact on supply chain: A framework for building smart, secure and efficient systems. *Futur. Gener. Comput. Syst.* **86**, 614–628 (2018)
 21. A. B. Mohamed, G. Manogaran, D. El-Shahat, S. Mirjalili, A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Futur. Gener. Comput. Syst.* **85**, 129–145 (2018)
 22. M. Abdel-Basset, G. Manogaran, D. El-Shahat, S. Mirjalili, A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Futur. Gener. Comput. Syst.* **85**, 129–145 (2018)
 23. A. B. Mohamed, M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah, S. Mirjalili, An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems. *Pers. Ubiquit. Comput.* **22**(5–6), 1117–1132 (2018)
 24. A. B. Mohamed, G. Manogaran, A. Gamal, F. Smarandache, A hybrid approach of neutrosophic sets and DEMATEL method for developing supplier selection criteria. *Des. Autom. Embed. Syst.* **22**, 257 (2018)
 25. M. Abdel-Basset, M. Gunasekaran, M. Mohamed, et al., *Neural Comput & Applic* (2018). <https://doi.org/10.1007/s00521-018-3404-6>
 26. M. Abdel-Basset, G. Manogaran, A. E. Fakhry, et al., *Multimed Tools Appl* (2018). <https://doi.org/10.1007/s11042-018-6266-0>
 27. M. Abdel-Basset, G. Manogaran, M. Mohamed, N. Chilamkurti, Threeway decisions based on neutrosophic sets and AHP-QFD framework for supplier selection problem. *Futur. Gener. Comput. Syst.* **89**, 19–30 (2018)
 28. V. Manisha, N. Bhardwaj, A. K. Yadav, Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment. *Int. J. Inf. Technol. Comput. Sci. (IJITCS)*. **8**(4), 1–10 (2016)
 29. M. Verma, N. Bhardwaj, A. K. Yadav, in *International Journal of Information Technology and Computer Science(IJITCS), Vol.8, No.4*. Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment, (2016), pp. 1–10. <https://doi.org/10.5815/ijitcs.2016.04.01>
 30. A. Yousefpour, G. Ishigaki, J. P. Jue, in *2017 IEEE International Conference on Edge Computing (EDGE), Honolulu, HI*. Fog Computing: Towards Minimizing Delay in the Internet of Things, (2017), pp. 17–24. <https://doi.org/10.1109/IEEE.EDGE.2017.12>
 31. B. F. Luiz, J. Diaz-Montes, R. Buyya, O. F. Rana, M. Parashar, Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput.* **4**(2), 26–35 (2017)
 32. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Experience.* **41**(1), 23–50 (2011)
 33. M. Aazam, E. Huh, in *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, St. Louis, MO. E-HAMC: Leveraging Fog computing for emergency alert service (IEEE, 2015), pp. 518–523. <https://doi.org/10.1109/PERCOMW.2015.7134091>

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
