Hindawi
Mobile Information Systems
Volume 2020, Article ID 8871976, 15 pages
https://doi.org/10.1155/2020/8871976

Research Article

A Lightweight Location-Aware Fog Framework (LAFF) for QoS in Internet of Things Paradigm

Qaisar Shaheen,[1,2] **Muhammad Shiraz**,[3] **Muhammad Usman Hashmi**,[2] **Danish Mahmood**,[4] **Zhu zhiyu**,[1] **and Rizwan Akhtar**[1]

[1]*School of Electronics and Information, Jiangsu University of Science and Technology, Zhenjiang, China*
[2]*Department of Computer Science, Superior College, Lahore, Pakistan*
[3]*Department of Computer Science, Federal Urdu University of Arts, Science and Technology, Islamabad, Pakistan*
[4]*Department of Computer Science, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, Islamabad Campus, Islamabad, Pakistan*

Correspondence should be addressed to Rizwan Akhtar; rizwan@just.edu.cn

Received 18 March 2020; Revised 13 August 2020; Accepted 29 August 2020; Published 16 September 2020

Academic Editor: Ali Kashif Bashir

Realization of Internet of Things (IoT) has revolutionized the scope of connectivity and reachability ubiquitously. Under the umbrella of IoT, every object which is smart enough to communicate with other object leads to the enormous data generation of varying sizes and nature. Cloud computing (CC) employs centralized data centres for the provisioning of remote services and resources. However, for the reason of being far away from client devices, CC has their own limitations especially for time and resource critical applications. The remote and centralized characteristics of CC often result in creating bottle necks, being latent, and hence deteriorate the quality of service (QoS) in the provisioning of services. Here, the concept of fog computing (FC) emerges that tends to leverage CC and end devices for data congestion and processing locally in a distributed and decentralized way. However, addressing latency and bottleneck issues for time critical applications are still challenging. In this work, a lightweight framework is proposed which employs the concept of fog head node that keeps track of other fog nodes in terms of user registrations and location awareness. The proposed lightweight location-aware fog framework (LAFF) persistently satisfies QoS by providing an accurate location-aware algorithm. A comparative analysis is also presented to analyse network usage, service time, latency, and RAM and CPU utilization. The comparison results depicts that the LAFF reduces latency, network use, and service time by 11.01%, 7.51%, and 14.8%, respectively, in contrast to the state-of-the-art frameworks. Moreover, considering RAM and CPU utilization, the proposed framework supersedes IFAM and TPFC targeting IoT applications. The RAM consumption and CPU utilization are reduced by 8.41% and 16.23% as compared with IFAM and TPFC, respectively, making the framework lightweight. Hence, the proposed LAFF improves QoS while accessing remote computational servers for the outsourced applications in fog computing.

1. Introduction

The concept of Internet of Things (IoT), supported by computational intelligence, has revolutionized in almost all domains of life. With every passing day, many new applications and domains in IoT and computational intelligence are emerging to help mankind in one way or other. On the other hand, providing such applications to general public has opened new horizons of business as well. In IoT, such businesses and applications mostly rely on the sensory data that has to be gathered for effective decision making. For fusion or manipulating big data (that may be some streaming data or in shape of batches), there are some requirements, i.e., distributed processing capability, effective communication and uncompromised network so that decision making process may yield better accuracy. Clouds being service providers tend to solve these problems. However, for the reason of being faraway from client

devices, they have their own limitations for time critical applications. Hence to reduce such complexities, the models of fog or edge computing are employed. Basic infrastructure for such environment comprises of *things* (computing devices) which have computing, communicating, and storing capability. Based on current trends, it is expected that by 2025 such smart environments will incorporate over 1 trillion IoT devices with 50% increased demand for latency sensitive applications [1]. Fog computing (FC) refers to a hierarchically distributed computing paradigm that bridges cloud data centers and IoT devices. The fog environment offers both infrastructure and a platform to run diversified software services. At different hierarchical levels of the fog environment, the physical devices are commonly called fog nodes. This technology overcomes the limitations of cloud computation by enabling data acquisition, processing, and storage at decentralized and locally available fog nodes [2]. The idea of this model is initially described by CISCO [3]. Figure 1 shows the general architecture of FC.

However, ensuring rich user experience (QoS) is the main concern to be addressed specially for time-sensitive applications such as health care IoT [4], web-based gaming [5], and video streaming applications [6]. The large distance between users and end devices increases the number of routers/hops which results in higher latency rate and network usage. Hence, real-time provisioning of services is obstructed and QoS is decreased while leveraging remote fog nodes for the outsourced applications.

In this work, we propose a lightweight location-aware fog framework (LAFF) which employs the concept of fog head node that keeps track of other fog nodes in terms of user registrations and location. The proposed LAFF persistently improves QoS by employing location-aware algorithm. LAFF addresses the issues of high latency, service time, and network usage in distributed data on fog server in order to improve the QoS. The location-aware algorithm involves user registration on fog head. The user/actuator is responded by analyzing their requested data types. Data types are divided into multimedia data (MMD) and textual data (TD). QoS through LAFF is compared with other contemporary frameworks [7, 8] to validate performance of the proposed framework. The significant contributions of this research are include the following:

(i) A fog-based lightweight framework is devised to provide better QoS to users

(ii) A location-aware algorithm is developed that enables latency reduction, service time reduction, and minimal usage of network resources

(iii) Resources utilization (RAM and CPU) is reduced to make the framework lightweight

The rest of the paper is organized as follows. The literature review is presented in Section 2. Section 3 discusses the lightweight location-aware fog framework (LAFF), location-aware algorithm, architecture, and analytical model. Section 4 focuses on the experimental setup of the framework. Section 5 presents the evaluation of the LAFF. Section 6 details the results of the simulations and discussion. The concluding remarks are conducted in Section 7.

2. Literature Review

In a simplified structure, FC is characterized by a geographically distributed computing design, prepared with heterogeneous devices connected at the edge of the network. The authors in [9, 10] highlight the advantages gained from FC. An algorithm is developed and implemented in [11] which is based on local computing. Through this algorithm, workload of cloud and fog processing is reduced. However, the proposed algorithm only works with star topology. In [12], a new layer is proposed, the fog layer (computing) of resources which is closer to the edge of the network to provide location awareness. A Fog-2-Fog (F2F) coordinated effort model is proposed in [13] that presents offloading approach amongst fog nodes, as per their load and handling capacities by Fog Resource Management Scheme (FRMC). In [14], the idea of resource allocation in a fog environment is introduced. The authors present a three-layer architecture that includes cloud, fog, and the user to divide the workload between the cloud and fog nodes. However, proposed architecture is only for homogenous environment. Moreover, scalability and associated challenges are not considered in the study.

Fan and Ansari [15] discussed the problem of load balancing in fog network through a distributed technique that assigns IoT devices to appropriate fog nodes and reduce the latency. In this technique, a fog node periodically broadcasts the computing and traffic estimated load. An FC framework is devised in [16] considering the medical field. Resource management is tackled by considering fog association, placement of VM, and task distribution. In [17], the workload placement algorithm is devised in tier edge cloud network to improve the response time of all tasks. The algorithm allocates computing resources between different tiers of fog node for completing assigned task. The idea of distributing the workload of a fog server receiving high traffic from IoT is presented in [18]. Two load balancing algorithms (task distributing and task grasping) are developed in [19] for large-scale FC. Through this structure, load balancing overhead is reduced when the scale of fog increased to get benefits of centralized and decentralized computing.

Puthal at al. [20] focus on developing an efficient dynamic load-balancing algorithm with an authentication method for edge data centers. Tasks were assigned to an underutilized edge data center by applying the breadth-first search (BFS) method. Each data center is modeled using the current load and the maximum capacity used to compute the current load. The authentication method allowed the load-balancing algorithm to find an authentic data center. IoT resource provisioning issue is discussed in [21], and a solution is proposed to overcome this problem. The model aims to boost fog resources and the minimization of system delay. The work in [21] is extended in [14] where QoS measurements and the deadlines for the provisioning of each kind of resources are considered.

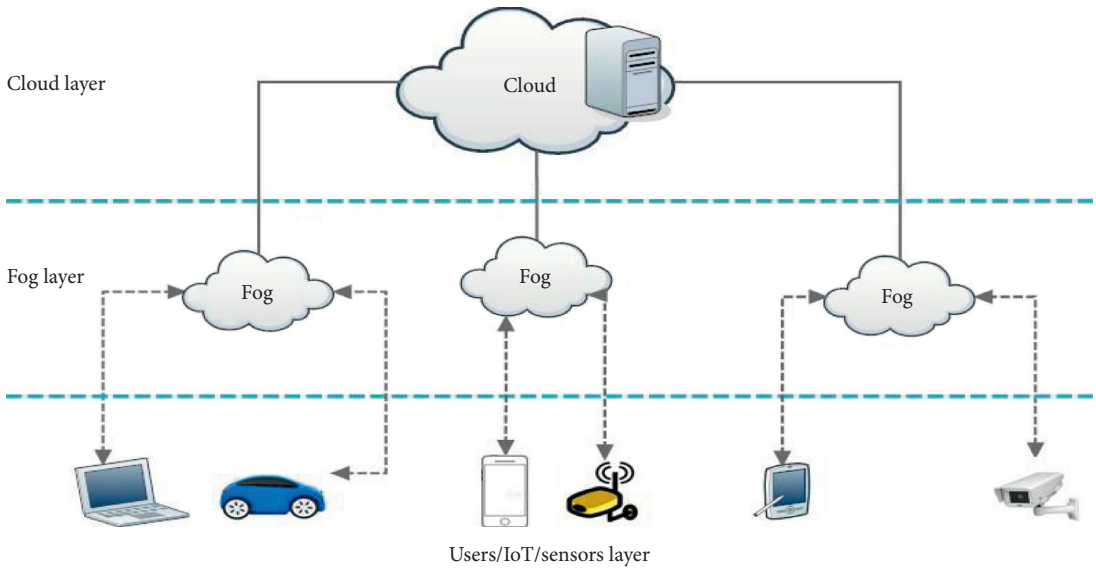

Figure 1: Architecture of fog computing.

In [22], a framework named FOGPLAN for QoS-aware dynamic fog service provisioning (QDFSP) is introduced. In order to meet low latency and QoS requirements of applications, QDFSP dynamically deploys application services on fog nodes or the release of application services that have previously been deployed on fog nodes. However, different characteristics of wireless and wired fog nodes are not considered. Also, the framework is neither location aware nor fulfils the real-time requirements of IoT tasks. Lin and Shen [23] designed a fog-based lightweight system to develop cloud gaming with high QoS. This system is a three-layer model, including cloud, fog, and devices (e.g., desktops/smartphone players). A set of supernodes were considered, which were near to end users and are connected to the cloud. The QoS requirements are achieved through reducing latency and bandwidth consumption.

A service management technique is introduced in [24] as iHome for smart house in the cloud. The paper proposed a service oriented architecture (SOA) to monitor home applications with real-time responses. The performance of services in terms of CPU and RAM in iHome is evaluated. The results show that the real-time responses can be returned under heavy burden of loading. The proposed system is tested under a limited number of physical appliances in a modular approach. However, many other important influential factors like cost and energy consumption are not addressed. Also, the system does not consider the user management and network condition. The proposed framework FATEH in [25] uses a three-layered architecture to improve QoS parameters. The first layer contains IoT devices and an agent node to collect data, and the gathered data are then submitted to the next layer. The third layer consists of fog manager to efficiently process the request on smart fog node. The processing and storage of less sensitive data are done at the third layer. The data coming from the fog manager are also processed at the third layer. The drawback of this system is that it does not take into account the network condition and user management.

An algorithm for task management in fog infrastructure is proposed in [26] aiming to focus on task scheduling at the fog layer while minimizing the response time dependent on resources requested by these tasks. In any case, explicit QoS prerequisites have not been considered in their methodology. Zeng et al. proposed an algorithm in [27] that works with a unified scheme for mobile IPV6 and suggests scheduling and handle user mobility. The issue of resource sharing among the fog nodes to execute computational requests was discussed, while they especially focused on fog-enabled little cells in cellular systems. In [28], Kim and Chung target the shaping clusters of small cells, where each cluster represents a collection of little cells that offer resources for offloading mobile devices from their remaining workload. The aim of this work is to reduce latency for each user through clusters shaping, bandwidth allocation, and computational resources.

Location-based services (LBSs) [29] become increasingly popular in recent years due to recent advancements in mobile computing. LBS refers to service provisioning through location-based information of users, i.e., the geographic position.

In [30], website performance optimization is automated by fogging at the edge servers. This idea explains the significance of edge location by giving dynamic and customizable optimization dependent on local network and the conditions of user's devices. WiCloud [31] is developed as mobile-edge computing platform with OpenStack to improve location awareness and to manage inter-mobile-edge communication and data acquisition for an innovative service.

Providing an acceptable level of QoS is an important issue in FC [32]. To design an efficient fog-based system, various QoS factors are considered. Extracting from the literature, eleven factors of QoS are defined, i.e., latency,

security, service time, availability, cost, energy consumption, resource utilization, reliability, execution time, deadline, and scalability [33, 34]. Moreover, latency is investigated as one of the important factors of QoS.

A framework is required to ensure QoS provisioning without burdening a single resource and provide service near to the edge focusing abovementioned performance metrics. This framework needs to be more useful to reduce latency, service time, and network use through user and location management by considering the network condition. A lightweight LAFF is devised through this study, and the framework possesses following characteristics.

LAFF has taken into account various IoT data requirements (multimedia data and textual data). Major emphasis in proposed framework is given to location awareness, i.e., knowing the exact location of the users/actuators. LAFF registers users on fog heads and employs K^* heuristic algorithm [35, 36] to find the shortest path between user and fog node. Moreover, the algorithm also takes the decision of fog head selection considering the requested data type.

The proposed work is compared with IFAM (intelligent FC analytical model) [7] and TPFC (task placement on fog computing) [8]. In [7], an analytical model and reinforcement learning algorithm in an FC environment is introduced. This model aims to reduce the latency among healthcare IoT, cloud servers, and end users. This paper proposes a novel multitier fog processing system that provides IoT services. However, in this work, the author did not consider user's location and network condition. The other drawback of this research is that user's request for normal data are transferred to the cloud to respond. The LAFF is better in terms that it considers user's location and network conditions. The framework also transfers both type of data, MMD and TD, to fog to fulfill user's request. In [8], a context-aware information-based approach ideally uses virtual resources accessible on the system edges to improve the presentation of IoT benefits in terms of response time, cost, and energy decrease. The approach utilizes context-aware information including network conditions location of IoT devices and service type to provide resources to IoT applications. However, the increase in the number of fog nodes and services causes an exponential increase in time for problem solving.

3. Location-Aware Fog Framework (LAFF)

In the proposed LAFF, location awareness under fog computing umbrella is introduced to reduce the latency, service time, and network usage along with minimal resources utilization. LAFF employs a location-aware algorithm that has ability to trace user's exact location through fog head. Fog head is the controller of data center of all fog nodes. The idea of fog head is used in fog computing technology [38]. The fog head node is not only limited to search for current nodes but also for new nodes ($\mathbf{F_{head}} \longrightarrow \mathbf{F_{MMD}} + \mathbf{F_{TD}} + \mathbf{F_{others}}$). F_{head} represents fog head node, F_{MMD} refers to the fog multimedia data node, F_{TD} is fog textual-data node, and F_{others} are nth new fog nodes. The search radius of fog head (F_{head}) is extended to nth new nodes as the framework is developed by keeping the idea of scalability as well. After accessing the user's exact location, fog head dedicates a nearest fog node in response to the user's request considering requested data type. If any nearest fog node is hard to reach, then the k^* algorithm is used to find the shortest path from user/actuator to fog node by estimating the coordinates [35, 36]. This dedicated node serves the user without any interruption. This framework also registers users (user management) and determines the requested data type. TD requests include text-based information, images, etc. (fog-TD servers handle these data types). MMD requests include videos, movies, etc. (fog-MMD servers handle these data types). The lightweight LAFF reduces the latency L_{ld}, service time f, and network use $\acute{u}_{\text{nw}}$. Figure 2 shows a detailed view of the LAFF.

3.1. Components of LAFF

3.1.1. Cloud Layer. The top layer of lightweight LAFF is a cloud layer which coordinates with lower layers for data collection and storage for future use. The cloud layer can be used for data processing and storage for a large amount of the data for longer duration. If the fog head fails to provide services to the user then cloud facilitates the users. Cloud layer components are as follows.

3.1.2. Cloud. Cloud is placed at higher layer of the lightweight LAFF. Cloud facilitates the fog layer in terms of storing data for later use and high processing when needed. Cloud servers are the centralized hosts. Cloud possesses all the necessary software needed to run, and it can also work as an independent unit. Cloud layer plays a supervisory role to handle communication and data storage. Cloud storage has many distributed resources acting as one unit. This distribution of data makes the cloud very fault tolerant. In this work, cloud is connected to the fog head to communicate with all fog nodes. Cloud communicates to fog head for all necessary communication. Cloud agent is responsible to manage communication between fog head and cloud.

The fog layer is the middle layer of the lightweight LAFF which aims to provide the processing facility of the data near to the edge. The following sections explain the modules of the fog layer:

3.1.3. Fog Head. Fog heads are fixed and predetermined physically with respect to geographical region and have lager hardware resources. Fog head is deployed between the fog nodes and the cloud and is responsible to communicate with cloud and all fog nodes. Fog head works according to the devised algorithm to access user's location and to identify requested data type. Users are registered at fog head. Fog head knows the exact location of all fog nodes. Tasks are assigned to the nodes considering the requested data type. Fog head is also responsible to manage and maintain the information on hardware level. Fog head has the following

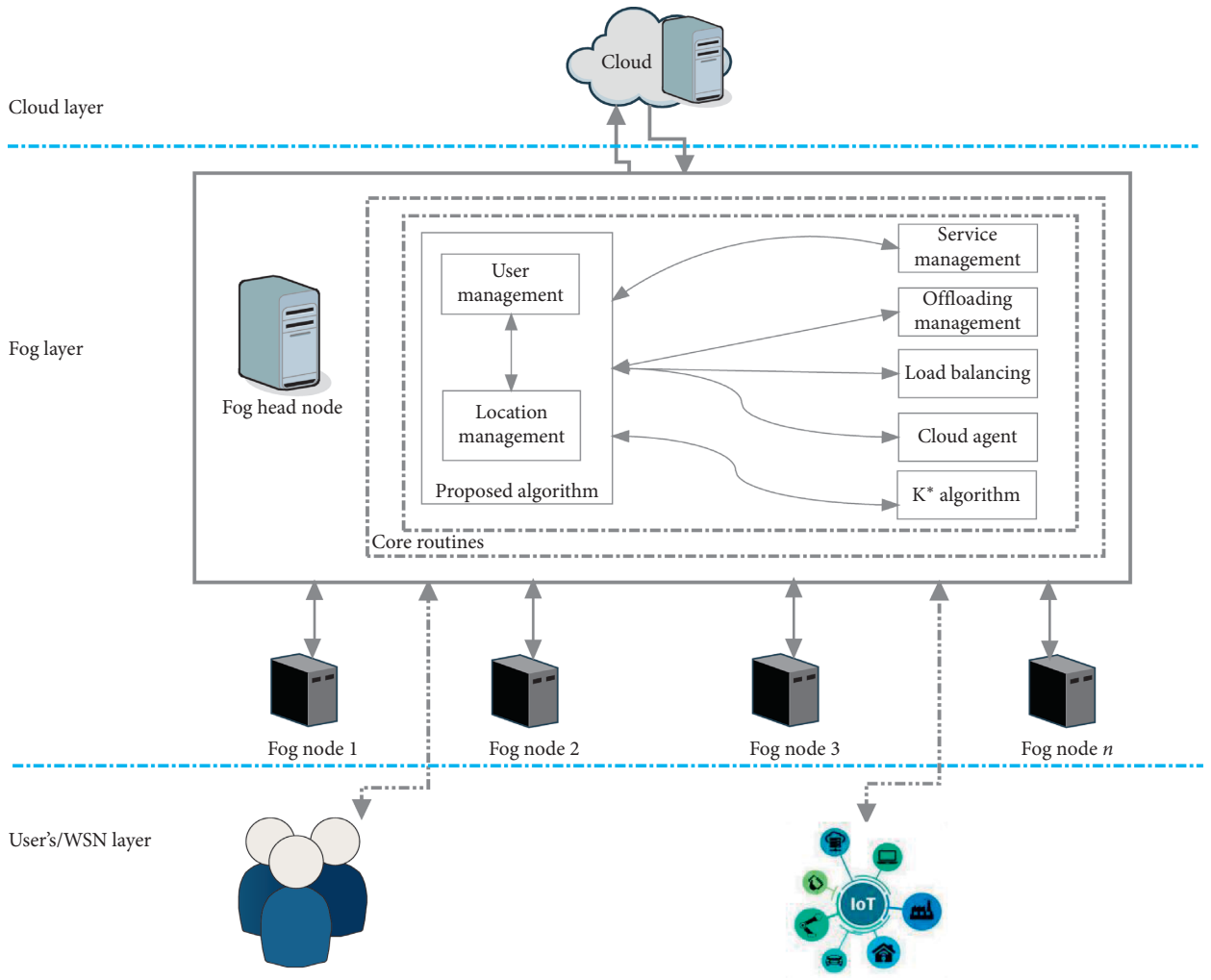

Figure 2: A detailed view of lightweight LAFF.

helping modules, and the proposed algorithm calls these modules as per their requirement.

3.1.4. User Management Module. Registration or management of the users and to store their details for future use is the responsibility of the user management module. The registered users are stored in Hashmap against specific identifiers for fast track communication. The advantage of using Hashmap is that it is not synchronized and hence saves additional usage of network and service time. The user management module communicates with the location management module to update/get user's location to provide services.

3.1.5. Location Management Module. The location management module manages the location of the users. The location is configured with coordinates. The x and y are range coordinate variables that are used for finding the shortest path in K^* search. The coordinates from 10 to 50 identify the location of existing users while other coordinates (coord1 and coord2) contain new users and n represents the coordinate value range. The mathematical representation of Geo function is described in the following equation:

$$\forall \mathbf{x} \cup \mathbf{y} \exists \mathbf{Geo}\,(\mathbf{coord}1, \mathbf{coord}2)\Delta \mathrm{coord}\theta < n \cap \mathrm{coord}\theta > n, \tag{1}$$

where $10 \leq n \leq 50$.

On each request of the user, the location management module is accessed to match/update the location management table.

3.1.6. Service Management Module. The service management module (SMM) provides services to the fog layer. SMM registers services and coordinates with fog nodes to provide services to users. It manages the fog node service delivery assurance. SMM monitors all the resources of the nodes and fog head.

3.1.7. Offloading Management Module. The offloading management module offloads the task from a fog node and assigns to other nodes to provide dedicated services to assure QoS. Through this module, the framework enables to offload the task from a fog node and dedicate to the user.

3.1.8. Load Balancing Module. The load balancing module distributes the fog layer traffic among different fog nodes. Through this module, the framework becomes more responsive and available for users.

3.1.9. Cloud Agent Module. The cloud agent module facilitates the fog head and cloud to communicate with each other for storing and updating data on the cloud. The cloud agent module works as a broker between the fog layer and the cloud layer.

3.1.10. Fog Nodes. Fog nodes work as a server of the geographic area in which the fog node is deployed. Fog nodes process data near to the edge to reduce the burden of the cloud. Through the fog nodes, the lightweight LAFF provides better QoS by reducing latency and service time to accomplish the request.

3.1.11. K^* Algorithm Module. Through the proposed algorithm, user's location is accessed and nearest fog node is assigned to the user to fulfill the request. If this nearest fog node is hard to reach due to any abnormality, this algorithm uses a heuristic search algorithm k^* [35], which is used to find shortest path between users and fog node. The vertices in this case are added between register and unregistered users. The advantage of using K^* algorithm is that it only uses the executed portion of the graph. It reduces the network usage by only working on a required portion instead of communicating to whole weighted graph. The complexity of K^* algorithm is $O(n\log n + r_u + u_u)$, where n is the number of vertices. In [39], Mishra et al. used the same k^* algorithm to find shortest path between source and destination.

3.2. Proposed Algorithm.

The LAFF algorithm is provided in Algorithm 1.

3.3. Features of LAFF. To minimize the service delay, fog head communicates to fog nodes and queries are processed on a short distance; in this way, the service latency is minimized. If queries are not communicated through fog nodes and transferred to the upper layer like fog head and cloud, then the service delays are at a larger value. The latency L_{ld} is calculated by dividing available time $T_{\text{available}}$ with total time T_{total} under product of 100. The following formula is used for calculating latency:

$$L_{\text{ld}} = \frac{T_{\text{available}}}{T_{\text{total}}} * 100\,(\text{milliseconds}). \tag{2}$$

IoT service delay-minimizing policy: policy adopted in this regard is to implement a minimum delay tolerance system. The values are considered to be very low as compared with that of other systems' latency. Latency, network use, and service time are reduced by using equation (1).

If a fog node is hard to reach, the LAFF uses k^* algorithm to find shortest path between users and the IoT devices. The path is selected from a pool of fog nodes (F_1–F_n) to have idle space for processing in order to provide better QoS. The list of fog nodes $F = \{F_1 + F_2 + F_3 + \cdots + F_n\}$ and users $U = \{U_1 + U_2 + U_3 + \cdots + U_n\}$ having tasks T for updating the Cloud C is represented by the following equation:

$$U_n \prod (F_n, T) \longrightarrow C. \tag{3}$$

Equation (3) represents the n array product from completion of task to update the cloud.

The remaining components of the proposed paradigm are mathematically defined in the analytical model. In the analytical model, we have discussed the mapping between the components of different layers. In the simulation, we implemented the analytical model in iFogSim.

3.4. Analytical Model. A set (S) T_{IoT} for all sensors S $\{S_1, S_2, S_3, \ldots, S_n\}$ and actuators A $\{A_1, A_2, A_3, \ldots, A_n\}$ under a tuples load α with transmission time L' is defined. Events E $\{E_1, E_2, E_3, \ldots, E_n\}$ happen at sensors $\{S\}$, where n is the nth mapped sensor to an event E. Equations (4) and (5) represent the events that happened at fog ℉ and cloud ℃ through sensors:

$$℉\colon \mathbf{T_{iot}}\{\mathbf{S_n}, \mathbf{E_n}\} * \boldsymbol{\alpha} \longrightarrow \mathbf{A_n}, \tag{4}$$

$$℃⊮ \mathbf{G}\colon \mathbf{T_{iot}}\{\mathbf{S_n}, \mathbf{E_n}\} * \boldsymbol{\alpha} \longrightarrow \mathbf{A_n}. \tag{5}$$

Within the increase in the number of hops amongst sensors, the latency, network usage, and service time also increase. The mapping of a sensor to a fog node is described in equation (6) that expresses the relationship between transmissions. Here, **i** is the IoT device number, **j** represents the column of devices where IoT device are mapped, and M is the mapping. L is the load (MMD or TD load), S is the sensor, and F is the fog node:

$$\mathbf{M}(\mathbf{i}, \mathbf{1})\colon \Sigma_{\mathbf{i=1}}^{\mathbf{n}}, \quad \mathbf{j} = \mathbf{1}L' * \mathbf{Si} \longrightarrow ℉. \tag{6}$$

The latency L_{td} is computed using equation (2).

The service time f is expressed in terms of time taken by a service provider SP $\{\text{SP}_1, \text{SP}_2, \text{SP}_3, \ldots, \text{SP}_n\}$ by providing a service $\check{T}$ to a user(s) $ú$ $\{U_1, U_2, U_3, \ldots, U_n\}$. The mapping relation is explained in

$$f\colon \mathbf{SP} \longrightarrow ú \prod \mathrm{L}' * \check{\mathrm{T}}. \tag{7}$$

To calculate the service time f in simulation environment, the following equation is used:

$$f = C_{\text{ins}}(T_{\text{ms}}) - T_k(S_t)\,(\text{ms}), \tag{8}$$

where C_{ins} (T_{ms}) represents the time in milliseconds fetched by calendar instance and T_k (S_t) is the simulation time stored by timekeeper class. The simulation time is the amount of time spent in processing the K^* search, allocating nodes, processing requests of users, and updating cloud related to processing. In order to calculate network usage $\mathbf{ú_{nw}}$, the tuple $\mathbf{T_{ud}}$ captured by network usage monitor $\mathbf{M_{nu}}$ are added to the total bandwidth used $\mathbf{B_u}$ in transmission and then divided by maximum simulation time $\mathbf{ST_{max}}$. the following equation is used for calculation:

Inputs: tasks T, start services S, user u, Geo (coord1, coord2) gets integer based coordinates.
Output: assign nearest Fog-MMD or Fog-TD to the user.

```
start;
submit tasks;
place operators;
start services;
while allusers do
  getlocation;
  if coord > 10 coord < 50 then
    existing reg user;
  else
    reg as new user;
  end
  if reguser then
    if multimedia data then
      if clocation = = plocation then
        if hard to find then
          start K* search;
          calculate tasks on nodes (F_n * T_n);
          offload data from nearest fognode (F * T − 1/T);
          allocate fog-MMD;
          F (u, T);
        else
          find nearest fog node;
          Search (F_1−>F_n);
        end
      else
        register location;
        Geo (coord1, coord2);
      end
    else
      transfer to fog-TD;
      F_l (u_u, T);
    end
  else
    unreg user;
  end
  find idle fog node;
  u_u (F_1−>F_n);
  send to cloud;
  u_u (C, T);
  repeat;
end
```

Algorithm 1: LAFF.

$$\acute{u}_{nw} = M_{nu}\left(T_{ud}\right) + \left(\frac{B_u}{ST_{max}}\right)(\text{Kbps}). \tag{9}$$

4. Experimental Setup

This lightweight LAFF is developed by conducting extensive simulation in CloudSim [40] and iFogSim simulators [41]. CloudSim is responsible for the simulation and events handling at Cloud. iFogSim handles events at Fog devices. This also minimizes the latency as servers become near to the edge of devices [42]. Following are the important steps and parameters which are needed to execute simulation.

The calendar is initialized to keep the current instance to conclude at the end when the simulation starts. In the end, the simulation variable is initialized by tracing flag to "false" so that the detail log which is not relevant to the simulation is not shown. The fog broker is initialized based on the data center broker. Considering the requirements of the clients related to QoS, the data center broker class coordinates between users and cloud service. A fog broker helps users to create tuples on the fog. Tuples are extended from cloudlets class to model tasks in CloudSim and iFogsim.

The cloud and fog data centers have their own characteristics. In real case, the characteristics of fog device are less powerful and have less storage than cloud data center. The capacity function of cloud $\complement(l+d)^n$ and fog $F(l)^n$ for load l and new expected data d is represented in equations (10) and (11):

$$\complement (l+d)^n = \sum_{k=0}^{n} \binom{n}{k} u^k d^{n-k}, \tag{10}$$

$$F(l)^n = \sum_{k=0}^{n} \binom{n}{k} u^k l^{n-k}, \tag{11}$$

where n is the number of total requests and k represents the capacity of responses that is sent against the requests n. The response k is always sent against request n. The $\complement (l+d)^n$ function equation (10) shows that the cloud has more storage than fog devices (equation (11)).

4.1. Cloud Data Centers. Cloud data centers (CDCs) are the centralized hosts and play a supervisory role to handle communication and data storage. Cloud storage has many distributed resources acting as one unit.

4.2. Fog Data Centers. Fog data centers store data for further processing and communication with users.

4.3. Location Manager Data Centers. Location management data store information regarding user's location.

4.4. Fog Head. Fog head knows the location of all fog nodes and communicates between the cloud and all nodes. Fog head is also responsible for managing and maintaining the information on the hardware level. Characteristics of fog data center, location manager, proxy server, fog head, Fog-TD, and Fog-MMD are given in Table 1.

4.5. Gateway Devices. These gateway devices are part of the fog layer and communicate with proxy server and cloud devices. Table 2 represents characteristics of gateway devices.

4.6. Sensor Devices. Sensor devices are created for scenarios which produce the data with following characteristics (Table 3).

4.7. Sensors and Actuators. As the actual device model is based on sensor devices, generating a huge amount of data that need to be processed, each device involves a sensor and an actuator attached to it. The purpose of the sensor is to "sense" the data which are identified by the selector module of the server.

4.8. Module to Module Interaction. Tuples are sent from one module to the other in order to interact with each other. The tuples which are sent up to the fog or cloud for processing are identified as TuplesUp and tuples that are sent downward from one module to the other are TupleDown. Also, tuples are mapped to modules using the tuple mapping techniques defined in iFogSim. The network usage is calculated on the basis of tuple flow. The network usage μ^n is defined in terms of μ^{fog} (fog network length) and μ^{cloud} (cloud network length) by dividing a tuple size T^L with simulation total time st as presented in the following equation:

$$\mu^n \longrightarrow \mu^{\text{fog}} + \mu^{\text{cloud}} \prod \frac{T^L}{\text{st}}. \tag{12}$$

5. Evaluation of the LAFF

The fog-based approach of the LAFF is shown in Figure 3:

Initially, the normal flow of the system is as follows:

User- > UserIdentifier- > ServiceHandler- > FogHead- > proxyServer- > Fog (MMD or TD)- > Cloud-server

The fog head handles user's requests. Through the location management module, user's location is traced and a fog head is deployed there to respond. If the location manager is not idle, then the proxy server can be formed. Fog head asks user identifier to identify the type of requested data. Requests may be for MMD or TD. After the fog head determines the type of the requested data, it allocates the required fog nodes to the users. The specific fog node facilitates the user accordingly. The fog-MMD node is loaded with very powerful processing capability, whereas a low spec is configured on fog-TD node. Table 1 represents the specifications of both fog-TD and fog-MMD. The proposed algorithm makes this work so unique and distinguishing.

The flow after initial one is given below:

User- > UserIdentifier- > ServiceHandler- > FogHead- > Fog (MMD or TD)- > User

If fog head fails to identify the relative fog service provider, the request is then transferred to the Cloud-server to facilitate the user as represented in Figure 4. The lightweight LAFF is a fault-tolerant framework due to the cloud's availability in case fog head fails to fulfill the request.

5.1. Data Configuration. A data set with tuple size 3000, bandwidth 1000, and network length 500 is implemented in the below mentioned configurations. The tuple in iFogsim represents the term data row, where there are sequences of bytes in such data rows.

Simulation runs on iFogSim for different configurations. The configurations are presented in Table 4.

The results of the abovementioned configurations are shown below.

5.1.1. Use Case. To prove the significance of the proposed algorithm, a use case is described.

5.1.2. Actors. Jeena, thief, and users (police vans) were the actors

5.1.3. Preconditions. Registered user with known location and requested MMD.

Table 1: Characteristics of devices.

Name of device	*MIPS	RAM	Uploading bandwidth	Downloading bandwidth	Level
Fog-data-center	20000	10 GB	10 Gbits	10 Gbits	1 (cloud child)
Location-manager	2000	1 GB	10 Gbits	10 Gbits	1
Proxy server	2000	2 GB	10 Gbits	10 Gbits	1
Fog-head	20000	8 GB	10 Gbits	10 Gbits	2
Fog-TD	20000	2 GB	10 Gbits	10 Gbits	3 (fog head child)
Fog-MMD	20000	4 GB	10 Gbits	10 Gbits	3 (fog head child)

*MIPS = million instructions per second.

Table 2: Characteristics of gateway devices.

Million instructions per second	RAM	Uploading bandwidth	Downloading bandwidth	Level
1000 Mips	1 GB	10 Gbits	10 Gbits	4

Table 3: Characteristics of sensor devices.

Million instructions per second	RAM	Uploading bandwidth	Downloading bandwidth	Level
1000 Mips	1 GB	10 Gbits	256 Mbits	4

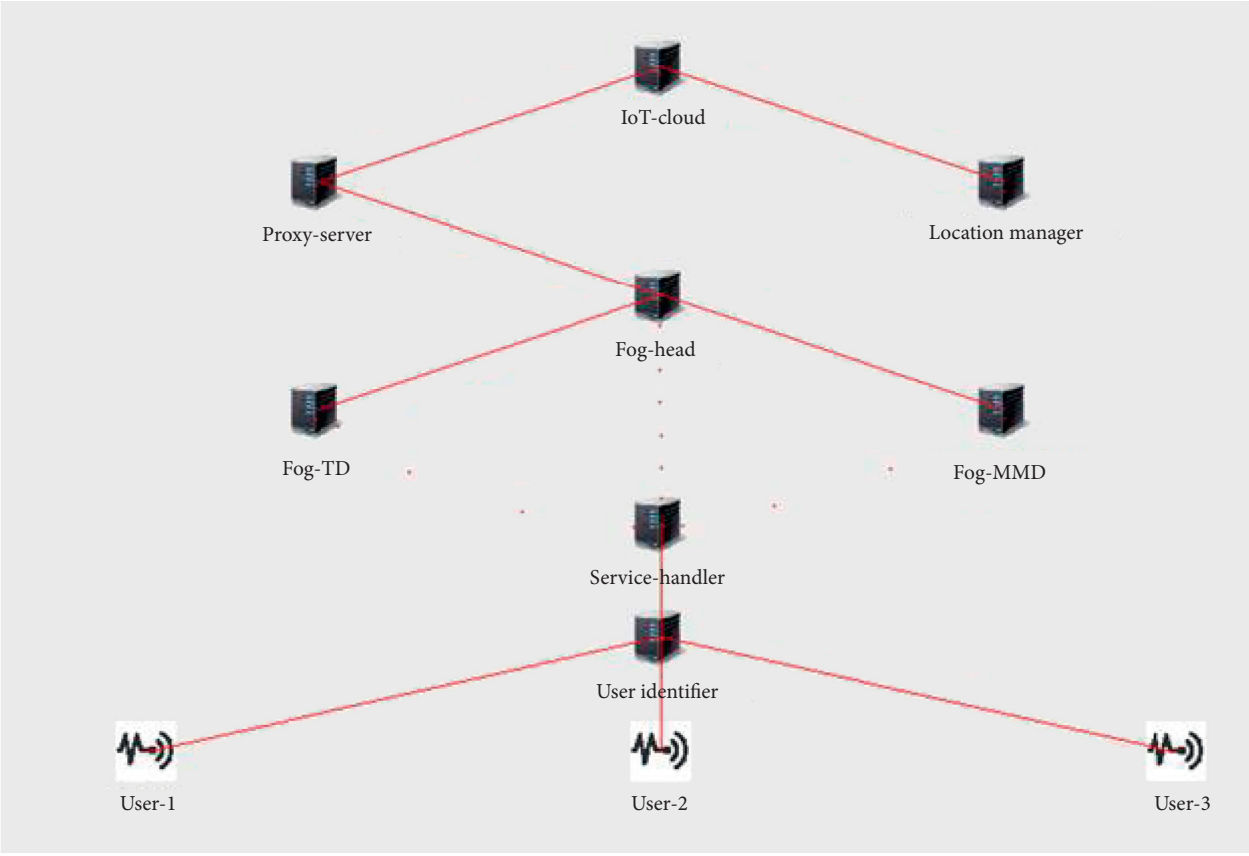

Figure 3: Topology of the LAFF.

5.1.4. Postconditions. A user is able to request fog framework to access CCTV cameras to get live streaming.

5.1.5. Scenario. Jeena is walking through a street; a thief snatched her bag and ran away. Jeena called the police and complained about the thief. The police man asked Jeena's location where she is now and to which direction the thief has gone. Jeena provides the police officer his desired information. The police officer started tracking the thief through CCTV cameras to get live streaming of the thief and also informed the police vans of the area where the thief is

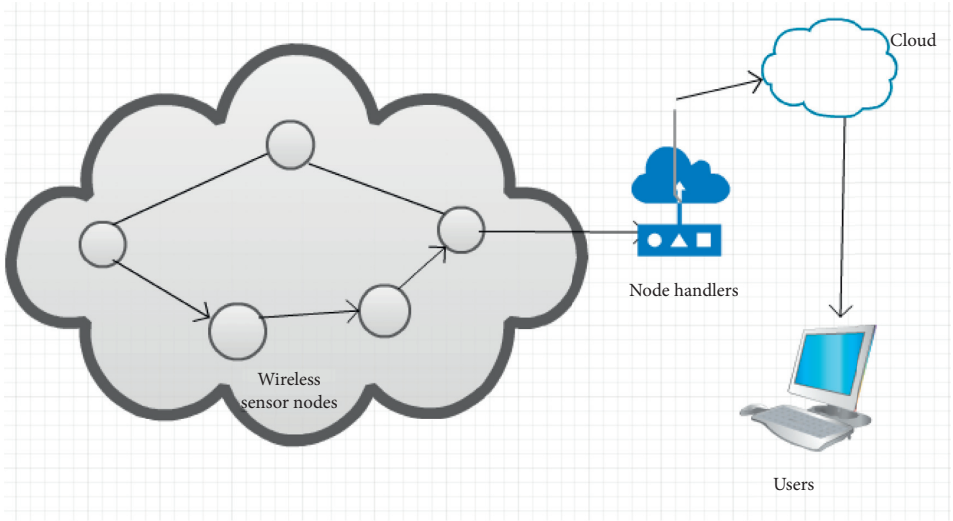

Figure 4: Flow of the LAFF in case when fog head fails to fulfil user's request.

Table 4: Data configuration.

Configuration	No. of fog nodes	No. of users
1	2	10
2	3	16
3	4	20
4	5	24
5	6	28
6	7	36
7	8	48
8	9	52
9	12	60
10	15	90

traced. The police vans caught the thief through accessing the exact location of the thief.

However, live streaming is a heavy task to run and requires a lot of computational power which requires a framework with low latency, service time, and network use to assure QoS. In this case, a nearest fog node will be assigned to the police vans so that they can trace the thief without any data loss and interruption.

6. Results and Discussion

The lightweight LAFF is compared with two other fog-based frameworks: IFAM (intelligent FC analytical model) and TPFC (task placement on fog computing) [7, 8]. The primary motivation behind this evaluation is to confirm the adequacy of the LAFF in terms of reducing latency, service time, and network use to facilitate users by providing better QoS. LAFF is a lightweight framework as it consumes less computational resources. RAM utilization and CPU consumption of a framework can increase the burden on resources. Since most of the fog nodes are not abundant in resources, execution of heavyweight software systems can cause significant computing overhead on them. Therefore, it is required to deploy lightweight frameworks in fog computing environments. The framework that consumes less RAM and CPU consumption is considered lighter than the other frameworks [43]. Ten configurations are employed with varying numbers of devices and nodes so that consistent patterns could be extracted.

6.1. Latency. Security applications are very time-sensitive. Results cannot be delayed. For instance, if we come to know that a terrorist is going to blast a bomb somewhere, finding the terrorist's locations on time is a crucial and time-sensitive task. Delay cannot be afforded as it can lead to very negative consequences. This delay is calculated by implementing a control loop. Latency is calculated by using a module to module latency, and then average of them is taken; latency is much higher when IFAM and TPFC modules are executed as depicted in Figure 5. This comparison is done in the established scenario for the LAFF. The results depicted that the lightweight LAFF reduced the average latency by 11.01% when compared with that of both the frameworks. The agenda not only stops at reducing the latency but also reduces the network usage and service time in order to provide better QoS and consistent data.

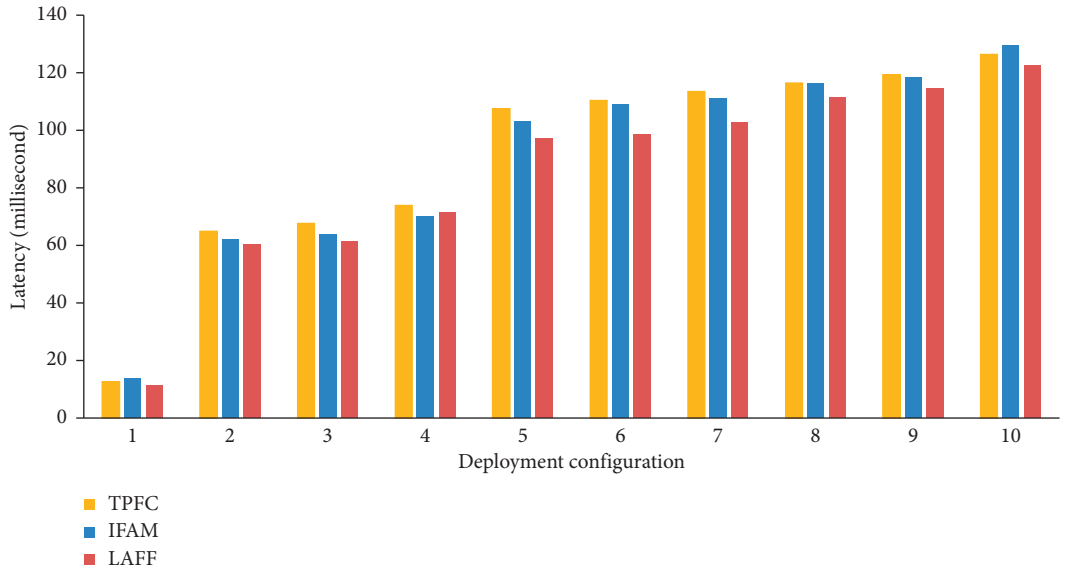

Figure 5: Latency comparison.

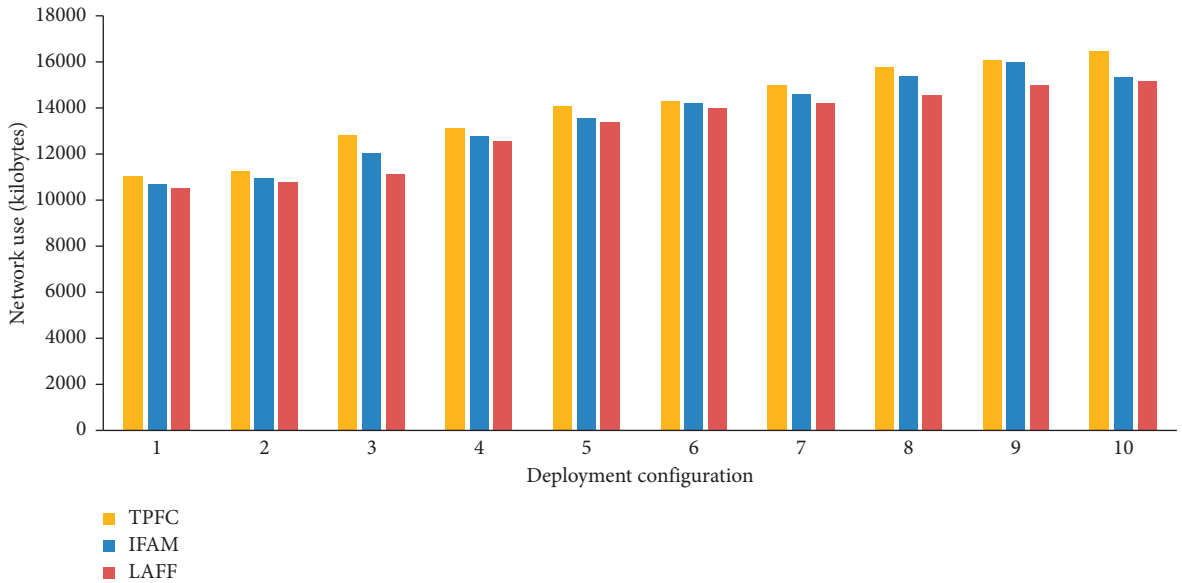

Figure 6: Network usage comparison.

6.2. Network Usage. This parameter is characterized by the utilization of system resources in terms of data sent and received from the network interfaces. The network usage should be kept at minimum for better performance. LAFF reduced the network traffic and consumption in terms of resource utilization. The results depict that the network utilization of the LAFF is reduced by average 7.51% as compared with that of the IFAM and TPFC as shown in Figure 6. The comparison of the LAFF with TPFC and IFAM showed that the LAFF provides better QoS.

6.3. Service Time. Service time is the most important parameter in sense of QoS. Service time is the amount of time spent to provide services to a user by a service provider. The service providers are the small hosts integrated with fog nodes and cloud in order to use storage and transmissions. The service time comparison is shown in Figure 7. It shows that the average amount of time is 14.8% lesser than that of the TPFC and IFAM.

6.4. RAM Consumption. RAM is one of the most important components of the fog node. If the framework consumes more RAM, the RAM system will crash and become unresponsive. To prove that the proposed framework is lightweight, RAM consumption of the framework is compared with that of TPFC and IFAM. Figure 8 shows the RAM

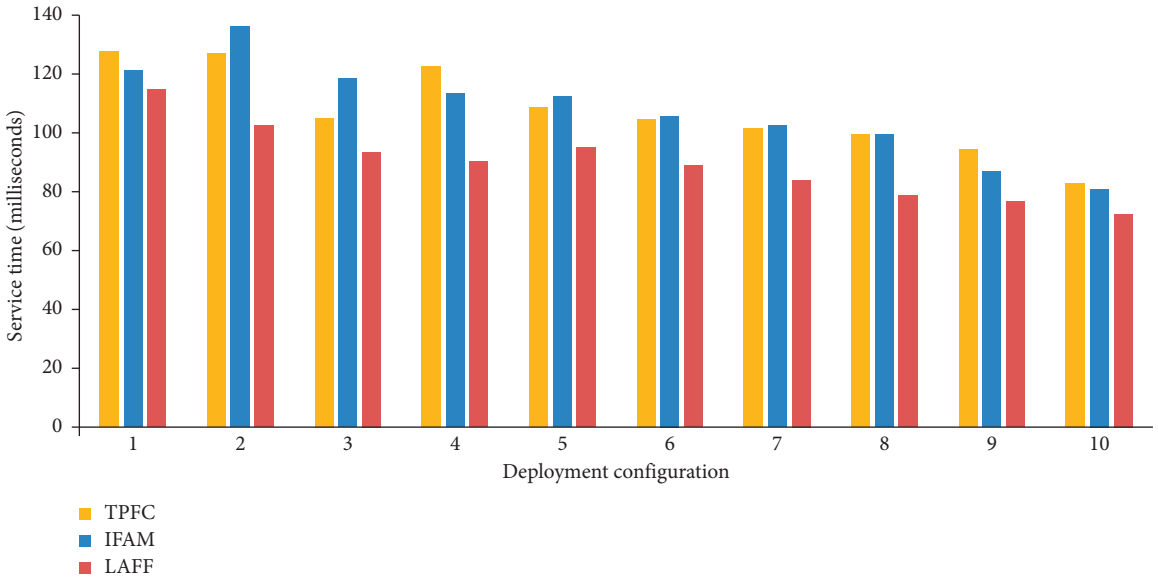

Figure 7: Service time comparison.

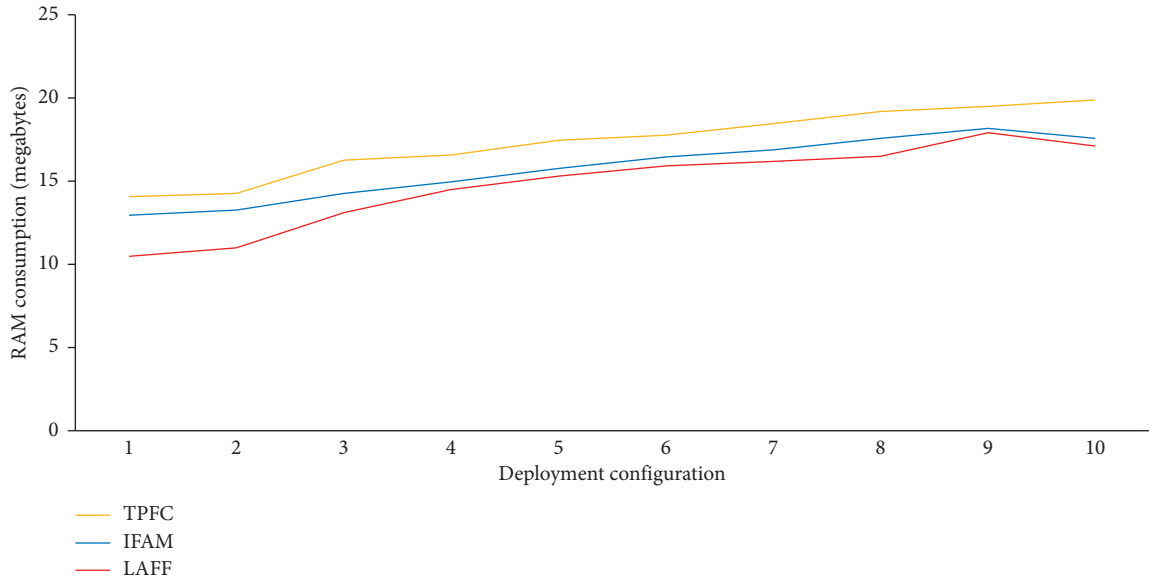

Figure 8: Ram consumption.

consumption for the data transmission and processing in fog nodes. The results showed that the proposed framework's RAM consumption is average 8.41% less than the both compared frameworks.

6.5. CPU Utilization. CPU utilization is the amount of work handled by a CPU of the fog node. The time taken between the start and the completion of a given task executed on a fog node is referred to as CPU utilization and measured in milliseconds. In this study, we do not include the time taken for separating and combining tasks before and after their scheduling. A task is composed of a set of instructions. We assume that each instruction requires one clock cycle to be executed. In the proposed framework, offloading module helps to minimize the CPU utilization, therefore increasing the fog node performance. The results in Figure 9 show that the proposed framework's CPU utilization is average 16.23% less than the both compared frameworks.

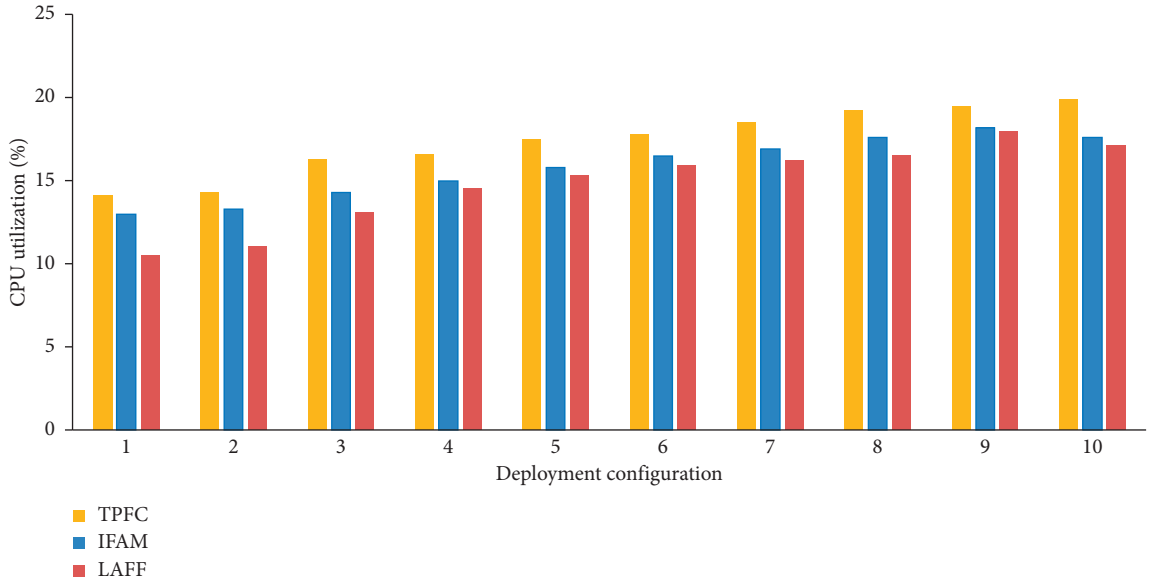

Figure 9: CPU utilization.

7. Conclusion and Future Work

The access to data and content is more smother and faster when accelerated with location awareness. Responsiveness and consistency are increased if the latency is minimized and bottleneck issues are catered. LAFF is designed as a location-aware algorithm which ensures rich user experiences and provides better QoS by reducing the network utilization, service time, and latency. It is examined that the LAFF lessens the average latency, network use, and service time by 11.01%, 7.51%, and 14.8%, respectively, as compared with those of IFAM and TPFC. Similarly, resource utilization in terms of RAM and CPU is reduced by average 8.41% and 16.23% as compared with that of TPFC and IFAM making LAFF comparatively a lightweight framework. Location-aware feature is significant in defense and intelligence areas. Hence, the proposed LAFF improves QoS while accessing remote computational servers for the outsourced applications in fog computing. For future work, it is suggested that a module for predictive analysis must be integrated in cloud which will be able to predict the user's requests by analyzing the user's location and previous requests time. We also plan to develop optimization mechanisms such as in [44] to determine the optimal distribution and configuration of fog nodes while taking into consideration the computational resources with a backup plan to provide the backup in case of system failures such as in [45] through introducing learning methods.

Data Availability

The data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the research project of the Natural Science Foundation of China (NSFC) under grant no. 61671222.

References

[1] McKinsey & Company, *The Internet of Things: How to Capture the Value of IoT*, McKinsey & Company, New York, NY, USA, 2018, https://www.mckinsey.com/featured-insights/internet-of-things/our-insights/the-internet-of-things-how-to-capture-the-value-of-IoT.

[2] National Intelligence Council, *Disruptive Civil Technologies: Six Technologies with Potential Impacts on US Interests Out to 2025*, CreateSpace, Scotts Valley, CA, USA, 2008.

[3] CISCO, *Fog Computing and the Internet of Things: Extend the Cloud to where the Things Are*, CISCO, San Jose, CA, USA, 2015.

[4] A. H. Sodhro, Z. Luo, A. K. Sangaiah, and S. W. Baik, "Mobile edge computing based QoS optimization in medical healthcare applications," *International Journal of Information Management*, vol. 45, pp. 308–318, 2019.

[5] S. Yan, M. Peng, M. A. Abana, and W. Wang, "An evolutionary game for user access mode selection in fog radio access networks," *IEEE Access*, vol. 5, pp. 2200–2210, 2017.

[6] S. F. Hassan and R. Fareed, "Video streaming processing using fog computing," in *Proceedings of the 2018 International Conference on Advanced Science and Engineering (ICOASE)*, pp. 140–144, Duhok, Iraq, 2018.

[7] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung, and A. Awang, "An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment," *PLoS One*, vol. 14, no. 11, Article ID e0224934, 2019.

[8] M.-Q. Tran, D. T. Nguyen, V. A. Le, D. H. Nguyen, and T. V. Pham, "Task placement on fog computing made efficient for IoT application provision," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 6215454, 17 pages, 2019.

[9] A. Munir, P. Kansakar, and S. U. Khan, "IFCIoT: integrated fog cloud IoT: a novel architectural paradigm for the future Internet of things," *IEEE Consumer Electronics Magazine*, vol. 6, no. 3, pp. 74–82, 2017.

[10] G. R. kumar, N. Saikiran, N. Saikiran, and A. Sathish, "FOG: a novel approach for adapting IoT/IoE in cloud environment," *International Journal of Engineering Trends and Technology*, vol. 42, no. 4, pp. 189–192, 2016.

[11] V. Mihai, C. Dragana, Grigore Stamatescu, and D. Popescu, "Loretta Ichim wireless sensor network architecture based on fog computing," in *Proceedings of the 2018 5th International Conference on Control, Decision and Information Technologies, (CoDIT'18)*, Thessaloniki, Greece, April 2018.

[12] M. Malensek, S. L. Pallickara, and S. Pallickara, "HERMES: federating fog and cloud domains to support query evaluations in continuous sensing environments," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 54–62, 2017.

[13] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via fog-2-fog collaboration," *Future Generation Computer Systems*, vol. 100, pp. 266–280, 2019.

[14] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," *International Journal of Information Engineering and Electronic Business*, vol. 8, no. 1, pp. 48–61, 2016.

[15] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered IoT," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 253–262, 2018.

[16] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 1, pp. 108–119, 2017.

[17] W. Tian, J. Zeng, Y. Lai et al., "Data collection from WSNs to the cloud based on mobile fog elements," *Future Generation Computer Systems*, vol. 105, pp. 864–872, 2017.

[18] I. M. Al-Joboury and E. H. Al-Hemiary, "IoT-F2CDM-LB: IoT based fog- to-cloud and data-in-motion architectures with load balancing," *EAI Endorsed Transactions on Internet of Things*, vol. 4, no. 13, 2018.

[19] C. Li, H. Zhuang, Q. Wang, and X. Zhou, "SSLB: self-similarity-based load balancing for large-scale fog computing," *Arabian Journal for Science and Engineering*, vol. 43, no. 12, pp. 7487–7498, 2018.

[20] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018.

[21] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog computing: principles, architectures, and applications," in *Internet of Things*, pp. 61–75, Elsevier, Amsterdam, Netherlands, 2016.

[22] A. Yousefpour, A. Patil, G. Ishigaki et al., "FOGPLAN: a lightweight QoS-aware dynamic fog service provisioning framework," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5080–5096, 2019.

[23] Y. Lin and H. Shen, "CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 431–445, 2017.

[24] G. Myrizakis and E. G. M. Petrakis, "iHome: smart home management as a service in the cloud and the fog," in *Advanced Information Networking and Applications*, pp. 1181–1192, Springer International Publishing, Berlin, Germany, 2020.

[25] S. Prabhdeep and K. Rajbir, "Design and develop quality of service framework using fog computing for smart city applications," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, no. 1S, 2019.

[26] Skarlat, O. Nardelli, M. Schulte, and S. Dustdar, "Towards QOS-aware fog service placement," in *Proceedings of the 1st International Conference on Fog and Edge Computing (ICFEC)*, Madrid, Spain, May 2017.

[27] D. Zeng, L. Gu, S. Guo, Z. Cheng, and S. Yu, "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3702–3712, 2016.

[28] W.-S. Kim and S.-H. Chung, "User incentive model and its optimization scheme in user-participatory fog computing environment," *Computer Networks*, vol. 145, pp. 76–88, 2018.

[29] M. K. Tefera, X. Yang, and Q. T. Sun, "A survey of system architectures, privacy preservation, and main research challenges on location-based services," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 6, pp. 3199–3218, 2019.

[30] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in *Proceedings of the 2013 IEEE 7th International Symposium on Service Oriented System Engineering (SOSE)*, pp. 320–323, San Francisco, CA, USA, March 2013.

[31] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: progress and challenges," in *Proceedings of the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pp. 83-84, Oxford, UK, March 2016.

[32] Y. Jiang, Z. Huang, and D. H. K. Tsang, "Challenges and solutions in fog computing orchestration," *IEEE Network*, vol. 32, no. 3, pp. 122–129, 2017.

[33] M. Haghi Kashani, A. M. Rahmani, and N. Jafari Navimipour, "Quality of service-aware approaches in fog computing," *International Journal of Communication Systems*, vol. 33, p. e4340, 2020.

[34] B. K. Dar, M. A. Shah, S. U. Islam, C. Maple, S. Mussadiq, and S. Khan, "Delay-aware accident detection and response system using fog computing," *IEEE Access*, vol. 7, pp. 70975–70985, 2019.

[35] E. M. Tordera, "What is a fog node a tutorial on current concepts towards a common definition," 2016, https://arxiv.org/abs/1611.09193.

[36] A. Husain and L. Stefan, "K^*: a heuristic search algorithm for finding the k shortest paths," *Artificial Intelligence*, vol. 175, no. 18, pp. 2129–2154, 2011.

[37] V. Santhanam and D. B. Shanmugam, "Integrating wireless sensor networks with cloud computing and emerging IT platforms using middleware services," *International Research Journal of Engineering and Technology*, vol. 5, no. 8, 2018.

[38] Z. Ning, J. Huang, and X. Wang, "Vehicular fog computing: enabling real-time traffic management for smart cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, 2019.

[39] M. Mishra, S. K. Roy, and A. Mukherjee, "An energy-aware multi-sensor geo-fog paradigm for mission critical applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 41, 2019.

[40] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of

resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[41] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and 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] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[43] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: a blockchain-based lightweight framework for edge and fog computing," *Journal of Systems and Software*, vol. 154, pp. 22–36, 2019.

[44] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proceedings of the 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, Lisbon, Portugal, pp. 1222–1228, 2017.

[45] M. K. Saroa and R. Aron, "Fog computing and its role in development of smart applications," in *Proceedings of the 2018 IEEE International Conference on Parallel & Distributed Processing with Applications*, Melbourne, Australia, December 2018.