International Journal of Computational and Experimental Science and Engineering (IJCESEN)

Vol. 11-No.4 (2025) pp. 8868-8884
https://www.ijcesen.com

ISSN: 2149-9144

Research Article

A mobility-aware service migration technique in fog computing environments

Saravjit Chahal[1*], Anita Singhrova[2]

[1]Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science and Technology, Murthal, India
* **Corresponding Author Email:** saravjitchahal.schcse@dcrustm.org - **ORCID:** 0000-0002-5247-6650

[2]Department of Computer Science and Engineering, Deenbandhu Chhotu Ram University of Science and Technology, Murthal, India
Email: anitasinghrova.cse@dcrustm.org - **ORCID:** 0000-0002-5247-7770

Article Info:

DOI: 10.22399/ijcesen.4319
Received: 10 April 2025
Revised: 28 April 2025
Accepted: 01 May 2025

Keywords

Fog computing
Migration
Mobility
Internet of Things
Real-time application

Abstract:

Fog computing allows the utilization of resources near the Internet of Things (IoT) devices to serve various latency-sensitive applications. However, the mobility of users of IoT devices necessitates the migration of applications to maintain service continuity and quality of service (QoS). This study proposes a new migration technique that minimizes delay, network usage and energy consumption in the Fog network, providing a real-time user experience. An objective function-based decision-making approach is used to migrate the applications efficiently, guaranteeing service continuity and QoS. The proposed technique chose an appropriate fog node with sufficient resources by evaluating parameters like connection duration between the fog nodes and the users, resource availability, and application execution time at the fog nodes. The results indicate that the proposed approach has a remarkable improvement of up to 20% in average delay, 16% in network usage and 7% in energy consumption compared to the conventional approach. The number of migrations is also lowered by 18%, which is necessary to efficiently utilize limited fog node resources as each migration event consumes additional resources. The benefits of the proposed approach for the users are low latencies, low network usage, improved energy efficiency and better user experience.

1. Introduction

A well-connected modern world is approaching where the Internet will be accessible to everything and everyone. According to a research report on Statista, there will be over 29 billion Internet of Things (IoT) connected devices by 2030 [1]. The sudden increase in IoT devices and internet traffic could lead to unforeseen disruptions if data processing isn't prioritized near the users. To address this, Fog computing has emerged as a solution to handle data processing closer to IoT devices. Due to their requirement to be compact, light, and battery-operated, these devices often have constrained hardware capabilities. Hence, they don't always make sense to host resource-intensive services directly. By positioning computing resources and services near the data source, fog computing extends and supplements the cloud, making it well-suited for supporting the IoT. Fog computing can involve any devices with computing power, including switches, routers, RoadSide Units (RSUs), smartphones, laptops, tablets and stationary equipment [2] [3]. However, fog computing is not restricted to IoT; it can also facilitate content delivery and support various other applications.Failing to finish the tasks for an emergency system application on time would lead to monetary losses. It could jeopardize human safety in scenarios like autonomous vehicles, emergency fire response, and emergency vehicle management. This cutting-edge computing paradigm encompasses all computing resources of the proximate network. It allows smart city, smart health and other IoT-driven Systems to execute the required applications close to the data source [4]. Fog proximity is the key enabler of many benefits that are not attainable when depending on cloud-based solutions. However, while Cloud is centralized in geographically remote data centers, Fog is deployed in proximity to IoT devices in a distributed way. Proximity in the topological distance is assumed, which is Which is determined

by counting the number of hops between the end user and the host [5]. The first and foremost of these benefits is a short communication distance that provides low latency between the host and the user. Other benefits of the close proximity of the Fog for the users are low bandwidth consumption, low latencies, better security and privacy, and uninterrupted services that remain available even when network connectivity to the Cloud is unreliable or completely unavailable [6]–[8].In the event of device mobility, maintaining low latency can be achieved by migrating the fog application across fog nodes along the device path. For a number of fog-based use cases, such as Virtual Reality and Augmented Reality applications performing video analytics, mobility support is crucial.Migration of applications across fog nodes is a significant and challenging task, resulting in additional overheads. When IoT users move around in fog environments, migration mechanisms determine when, how, and where applications can migrate.Containers, a lightweight virtualization technology, may be preferred over virtual machines to host the applications to guarantee minimal network overhead [9]. The data and applications pertaining to users are encapsulated in the containers [10]. The new developments in fog computing confirm that containers perform superior to conventional virtual machines [11].The following factors affect service migration operations: the IoT device's position, direction and speed, and finding an appropriate node to transfer the application. Based on these parameters, the Fog environment's performance metrics, such as network delay, bandwidth consumption, etc., can also vary significantly [12]. The migration process comprises moving data and the associated application content encapsulated in the containers. The methods that keep migrating the applications along the path of users result in many undesirable migrations. Every migration request invites an overhead in the fog environment with limited fog resources. Thus, the migration is performed when it is impossible to prolong the execution further. The application may remain on the same Fog node if there are no feasible options to migrate the application.In light of the above, a migration technique is proposed to fulfill the needs of real-time mobile IoT applications. In summary, the contributions of this study are as follows:

1. A mobility-aware application migration algorithm is proposed to decide where to migrate applications along with the migration point.
2. The effectiveness of the proposed algorithm is validated through extensive simulations utilizing mobility traces. A comparison with an existing well-established algorithm reveals that the proposed method significantly improves delay, network usage and energy consumption.

The rest of the paper is structured as follows. Section 2 presents the relevant existing work of migration management techniques in fog computing. Section 3 describes the proposed system model and problem formulation, followed by the proposed algorithm in section 4. Section 5 discusses simulation results of the proposed approach and compares it with traditional techniques. Finally, section 6 summarizes the article and outlines potential future research directions.

2. Related Work

This section discusses some conceptual and fundamental work in related areas.The authors of [13] proposed a hierarchical Fog computing architecture. The data generated by the end devices was forwarded to the fog nodes and not the Cloud. The end devices or the users demanding the services might be mobile in nature. One of the essential characteristics of the fog system was having mobility support. Preserving continuity in providing service at different locations was a tough job. The study in [14] proposed a model for migrating virtual machines aimed at Mobile Cloud Computing environments. Their technique was built on the cloudlet load and user mobility parameters. A genetic algorithm was used to find a suitable server and reduce the frequency of migrations.The authors emphasized the necessity of mobility-aware scheduling and put forward a solution based on the edge ward placement method for Fog computing environments in paper [15]. They highlighted the important indicators to take into account in the Fog environment supporting user mobility.

Table 1. *Study of various papers discussed*

Reference	Main Idea	Parameters considered	Achievement	Weakness
[14]	Load balancing for heterogeneous mobile cloud computing using Genetic algorithm-based solution	number of migrations, task execution time	number of migrations reduced, Avg. task execution time reduced	VMs consolidation not considered

[15]	Resource management using various scheduling strategies in fog computing	Latency, execution cost, network usage	Latency reduced	User mobility not considered
[16]	Resource management in Fog and Edge computing environments through various placement policies	Latency, cost, network usage	Latency reduced, network congestion reduced, energy consumption improved Cost decreased	User mobility and container migration not supported
[17]	ILP based resource allocation in fog environment	Latency, QoS	Latency reduced, QoS improved	Network usage and energy consumption not considered
[18]	Task scheduling in fog nano data centres utilizing container virtualization technology	Energy consumption, SLA violations, response time, makespan	Energy consumption reduced, SLA violations reduced	Latency not considered
[19]	Resource allocation for mobile micro-clouds based on polynomial cost functions	Cost	Minimized average cost	Heterogeneous resources in cloud not considered
[20]	Route optimization in mobile fog computing	Latency, handover performance, data communication, system cost	Latency reduced, handover performance improved, system cost reduced	Energy consumption and network usage not considered
[21]	Resource allocation in Mobile Edge Computing environments.	migration time, downtime	overall migration time reduced, downtime reduced	Performance under large-scale networked MEC systems not considered
[22]	Resource optimization in heterogeneous fog computing environments using linear programming	Latency, application placement time, service delivery latency	Latency improved	Real-world implementation not considered
[23]	Application deployment in fog computing using linear optimization and Fuzzy logic	Cost, Packet loss rate, network usage, quality of service	Latency and resource consumption reduced, network usage reduced, service quality improved	Performance in real Fog environment not considered
[24]	Service placement in fog computing	Latency, hop count, network usage	Improvement in network usage and latency	Degradation of service for less requested applications observed.
[25]	Resource utilization using Analytic Hierarchy process (AHP)	received signal strength, user velocity, data rate, signalling cost	ping pong rate reduced, throughput improved, packet delay reduced, signalling cost reduced	Real-world implementation not considered
[26]	Mixed Integer Linear Programming (MILP) task allocation for vehicular Fog computing	Latency, quality loss and Fog capacity	Latency reduced	Not feasible for large-scale deployment of Fog nodes and users
[27]	Load balancing in vehicular Fog computing environment utilizing the Simulated Annealing Algorithm (SAA)	Resource utilization	Resource utilization improved	Load balancing not achieved under limited computing resources
[28]	Resource management in Fog/Edge computing	Latency, execution time, received signal strength response time, SLO violations	Low latency achieved, SLO violations minimized	Context of multiple competing IoT applications not considered
[29]	Task offloading in Fog environment using Gini Coefficient and GA	Migration cost and energy cost, sojourn time,	Reduced migration time and enhanced revenue for user equipment	Migration cost remains high
[30]	Resource allocation in Fog computing using heuristic search	Latency, resource availability, throughput, energy consumption, jitter	Latency improved, energy consumption decreased	Collaboration between edge and fog devices across different regions was not taken into account

[31]	GA based resource allocation in Fog computing	Loop delay, network delay, execution cost, network usage, execution cost	Latency improved, network usage reduced, execution cost reduced	Mobility prediction technique was not used
[32]	Task offloading using machine learning in fog environment	Network usage, migration time, number of migrations, number of handoffs	Latency reduced, energy consumption reduced, network usage decreased	Inefficient for large-scale systems
[33]	Resource allocation in mobile edge computing leveraging Reinforcement learning and Markov decision process (MDP)	Migration delay, computation delay, communication delay, migration cost	Latency improved, migration cost reduced	Load balancing among microservices is not considered
[34]	Resource optimization for E-healthcare applications using Markov decision process (MDP)	Resource usage cost, migration cost, reconfiguration cost	Total cost reduced, achieved maximum expected reward	Not suitable for large-scale networked MEC systems.
[35]	Resource allocation in fog computing	Latency, processing time, SLA violations	Latency minimized, processing time reduced, SLA violations reduced	Failure of fog nodes not studied
[36]	Application deployment in Edge and Fog Computing Environments using weighted cost model	Response time, energy consumption, total Migration cost, number of interrupted tasks	Average execution cost reduced, cumulative migration cost reduced	Energy consumption of servers and monetary cost not considered.
[37]	IoT Service Placement in Fog Computing based on Open-source Development Model Algorithm (ODMA) metaheuristic	Latency, energy consumption, service cost, Fog resource utilization	Latency reduced, resource usage improved, service acceptance rate improved	Reliability and safety of interactions not considered
[38]	Service Placement for utilization of fog resources using evolutionary algorithm based on the cuckoo search	Latency, response time, energy consumption, SLA violation, communication cost, computation cost, Fog utilization	Latency reduced, energy consumption decreased, fog utilization improved	Reliability and fault tolerance not considered
[39]	Service deployment in fog-cloud environments based on genetic optimization	Network utilization, latency, energy efficiency, execution cost	improvement in latency, network utilization reduced, energy efficiency enhanced, cost reduced	Dynamic requirements of IoT applications not considered

User mobility patterns and application priority levels were accounted for in making effective scheduling decisions. Though, user mobility was not considered in the existing scheduling approaches. The authors focused on placing IoT applications while taking into account their target location [16]. The possibility of clustering was not taken into account in the proposal. The application modules were therefore sent to the next hierarchical tier for potential migration and placement whenever the existing server was unable to serve the application modules.The study in [17] extended the iFogSim simulator to add mobility support. The authors designed migration strategies for mobile users. The container virtualization technology improved performance compared to conventional virtual machines [18]. The study in [19] offered a technique for deploying a single service instance for each IoT user on a distant server when several IoT users were present in the system. To identify optimal and nearly optimal solutions, they introduced offline and online approximation techniques for the Cloud.The study in [20] proposed mobility-supported Fog computing architecture. The Software-defined networks-based architecture was proposed to decouple mobility control and data forwarding. A framework was designed to facilitate mobility in Mobile Edge Computing environments (MEC) [21]. The authors provided the service without interruption and migrated services across MECs. Their approach was aimed at lowering downtime and overall migration time. Many researchers devised techniques for the initial placement of services in heterogeneous contexts. Some of these methods were designed to lessen the delay in service delivery [22] and increase users' quality of experience [23]. The placement solutions were either centralized or decentralized [24].The study in [25] proposed a handover strategy in wireless communication technologies for mobile users. The authors presented a multi-criteria handover strategy for mobile users based on various parameters to avoid unnecessary handovers and improve the utilization of resources. The proposed strategy significantly brought improvements in various QoS parameters. The authors of [26] and [27] studied vehicular Fog computing environments.

The study in [26] aimed to optimize vehicular Fog computing-based task allocation. Various constraints like quality loss, latency, and Fog capacity were considered for modeling optimization of the task allocation problem. However, the technique was not viable for scenarios involving a vast number of fog nodes and users in the system. In contrast, a fog-enabled mobility-based migration framework was proposed for smart cities in [27]. The load balancing was achieved among fog nodes according to a resource pricing-based incentive strategy. Limited resource capacity confined the extent of load balance attained.The placement strategies for IoT applications with many interconnected modules that take into account historical mobility data were suggested in [28]. The authors presented a cloud-centric method termed URMILA, whereby the placement of all IoT applications was decided by a centralized controller to meet their latency needs. Also, there was no migration mechanism to transfer the applications to new servers if the user had moved outside the range of its existing server, which resulted in a considerable cost to the user. The authors of [29] proposed a mobility-aware strategy for offloading and computational resource allocation that significantly reduced migration times. The main goal of the work was to lessen the number of migrations while maximizing offloading benefits for IoT users.The new advances in the Internet of Everything (IoE) demanded real-time execution of service requests [30]. The fog nodes closer to the end user enabled real-time response, fulfilling the requirements of real-time applications. An autonomic hybrid framework was proposed to perform container migration [31] while satisfying the QoS requirements of the user. A mathematical model was developed to predetermine the target node for migrating the user module. However, more precise techniques, such as mobility prediction modes, were not utilized to anticipate the user's future location. The authors of [32] focused on IoT scenarios and proposed a learning-based fog node selection scheme demanding extremely low latency. They introduced a mapping function to offload the task to a suitable fog node. The proposed system predicted the location of IoT devices using machine learning-based methods.The authors of [33] introduced edge-centric application deployment and mobility management techniques when there were many IoT users in the system. The authors' primary objective was to minimize service delay. The work in [34] analyzed that migration incurred communication and computation overheads. Thus, the decision on migration depended on multiple factors, including user mobility, and the availability of resources in heterogeneous edge clouds.The work in [35] proposed a resource allocation technique based on multiple criteria to choose a suitable resource for the execution of a real-time task in fog environments. The work considered dynamic user behavior after application submission but did not study the failure of fog devices. The study in [36] proposed a weighted cost model for reducing device energy consumption and response time. The authors also proposed a clustering method that allowed for the cooperative execution of tasks and provided improved services for the applications. A migration management technique that reduced the migration cost of IoT applications was also presented. The recent studies in [37] and [38] aimed to optimize service placement policy for efficient resource utilization and improved QoS. The authors of [37] proposed an autonomous method for service placement based on a conceptual framework presented in the same study. While the authors of [38] have prioritized the requests for optimal service placement to enhance the performance concerning various metrics, considering the heterogeneity of resources and QoS deadlines of applications. The authors of [39] have focused on reducing the network usage and application delay by proposing a genetic optimization-based module placement algorithm. They have introduced a penalty-based method to reduce the delay. In the proposed algorithm, authors considered different factors, including communication delay between modules and their hierarchy level in the network. The discussion above makes it clear that the authors have strived to reduce the delay experienced by the users, energy consumption, and network usage in the Fog environments. The authors have put effort into reaching these goals, even though they have considered only fog nodes resources characteristics and worked towards assigning the nearest fog node for the migration of applications. Each migration event consumes additional Fog resources; thus, the migration count should be lowered. The authors have not worked significantly in this direction. These methods keep migrating the applications, resulting in many undesirable migrations inviting overheads in the Fog environment. Fog nodes have limited processing capability; therefore, efficient utilization of available resources is essential to enhance user experience.Moreover, the rise in the number of IoT devices is exponential and will require abundant resources. It is evident that no one has considered the Fog environments' distinctive characteristics, such as execution time, mobility direction, and contact duration of IoT users with Fog nodes altogether. In this research, in addition to the above factors, resource requirements and execution time have been considered in the proposed strategy to address the migration issue in the Fog

environments. An effective approach for migrating the applications without producing overhead in the network has been proposed. This is achieved primarily by reducing the number of migrations in the system.

3. System Model and Problem Formulation

A three-tier hierarchically organized architecture consisting of Cloud, Fog, and IoT devices are considered, as shown in the mobility scenario depicted in Figure 1. The topmost layer comprises of Cloud and is used for processing and storing the data. The fog layer is located near the user and is assumed to have several fog nodes. This layer provides services to mobile users and is responsible for executing the applications. The fog nodes use a lightweight container virtualization technology to deploy application modules. The Fog scenario is considered to deliver services in real-time for users whose location is dynamic.

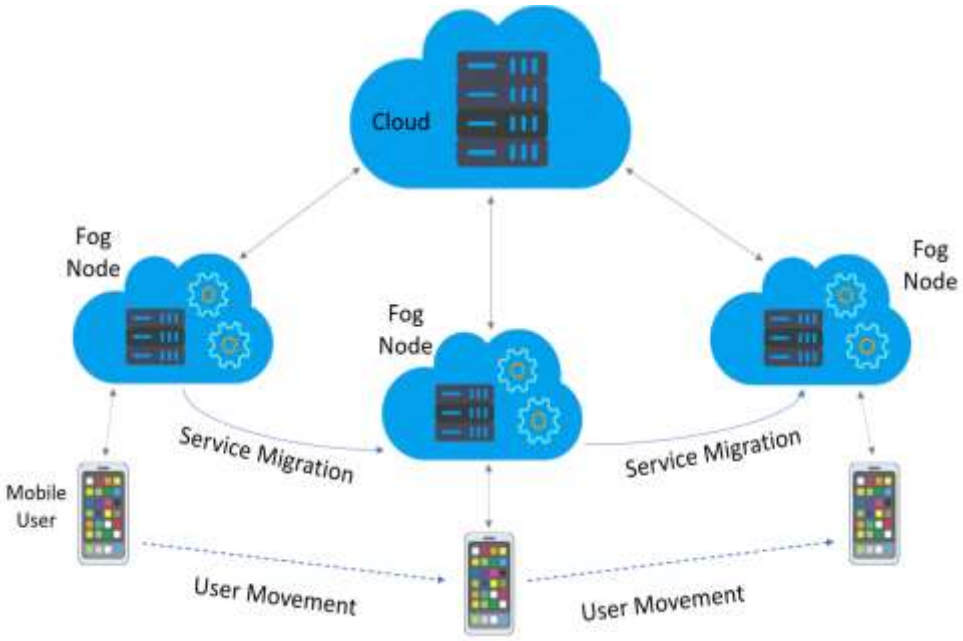


Figure 1. *Mobility scenario: service migration in Fog environment*

A set of fog nodes $F = \{f_1, f_2, f_3, \ldots\ldots, f_m\}$ is considered where fog nodes are interconnected via wired or wireless links. It is assumed that fog nodes have heterogeneous resources from a hardware architecture point of view; thus, computational power is not similar. Fog nodes handle tasks like managing user requests, resource allocation, and application migration. It is also responsible for deciding on migrations and migrating the live applications among the fog nodes. The data that cannot be handled at the fog layer and the data needed for future examination may be sent to the Cloud.The computational capacity of a fog node, f_i is given in units of million instructions per second (MIPS) because this is how MobFogSim [5] represents the execution capacity. Fog nodes are defined based on their resource capacities. These resources are computational capacity, memory, and bandwidth. C_i^{cap} is the computational capacity of a fog node, f_i. M_i^{cap} is the memory capacity, and B_i^{cap} is the amount of available bandwidth on the same fog node, f_i. The key notations employed in the system are listed in Table 2. The users of IoT devices are deemed to have mobility, and these have mobility timelines or direction and speed. User applications are executed on suitable fog nodes. Ideally, this may occur at the fog node connected to the user. When mobile users

Table 2. *Table of key notations*

Notations	Description
F	Set of Fog nodes in the system
M	Number of fog nodes in the system
f_i	The i^{th} fog node in the system
A	Set of application modules in the system
N	Number of application modules in the system
a_j	The j^{th} application module in the system
C_i^{cap}	The computing capacity of the fog node, f_i
M_i^{cap}	Memory capacity of fog node, f_i
B_i^{cap}	The available bandwidth on the fog node, f_i
P_i	Amount of computing required by application module, a_j
R_i	Amount of memory required by application module, a_j
T_i	Amount of bandwidth required by application module, a_j
x_{ij}	A binary variable to determine whether $a_j$is assigned to f_i

change their location, the data and the application need to be migrated promptly to maintain service continuity and a certain Quality of Service (QoS) in the network. The goal of the migration decision is to choose an appropriate destination fog node amongst multiple available fog nodes for receiving the application modules of each application to lower the response time of application modules. Consider a set of IoT applications, $A = \{a_1, a_2, a_3, \ldots, a_n\}$ in a dedicated geographical region such that each user is connected to a suitable fog node**.** Different fog nodes serving users deploy the application modules as containers. Each application is assigned to an appropriate fog node having sufficient resource. This research addresses the issue: How can latency requirements be met considering the mobile user needs for real-time applications considering the migration of applications to a suitable fog node? It needs to have some decision-making for the efficient application migration to optimize the overall QoS. In this paper, latency is considered the key QoS parameter. The question is, for the application described above, which fog node would be more appropriate?The migration decision-making problem is formulated as maximizing the objective function shown in Equation 1. The QoS objectives are taken into consideration when modeling the objective function.Maximize

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \left(Ct_i^j - Et_i^j\right) * x_{ij} \quad (1)$$

where Ct_i^j is the estimated connection duration of the user's application module, a_j with the candidate destination fog node, f_i and Et_i^j is the estimated execution time of the application module, a_j on the candidate destination fog node f_i. m and n are the number of fog nodes and application modules in the system, respectively, and, x_{ij} is a binary decision variable used to determine whether a_j is assigned to f_i or not.

Subject to the following:

$$\sum_{i=1}^{m} x_{ij} \times P_i \leq C_i^{cap} \ \forall j \in \{1, \ldots, n\} \quad (2)$$

where x_{ij} is a binary decision variable, P_i is the amount of computing required by the application module, a_j, and C_i^{cap} is the computing capacity of the fog node, f_i.

$$\sum_{i=1}^{m} x_{ij} \times T_i \leq B_i^{cap} \ \forall j \in \{1, \ldots, n\} \quad (3)$$

where x_{ij} is a binary decision variable, T_i is the amount of bandwidth required by the application module, a_j and B_i^{cap} is the bandwidth available on the fog node, f_i.

$$\sum_{i=1}^{m} x_{ij} \times R_i \leq M_i^{cap} \ \forall j \in \{1, \ldots, n\} \quad (4)$$

where x_{ij} a binary decision variable, R_i is the amount of memory required by the application module, a_j is and M_i^{cap} is the memory capacity of the fog node, f_i.

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall j \in \{1, \ldots, n\} \quad (5)$$

where x_{ij} is a binary decision variable.Equation 2 ensures that the computing requirement of a set of application modules allocated to the fog node, f_i should not surpass the computing capacity of the fog node. The constraint in equation 3 specifies that the bandwidth needed to execute a set of application modules assigned to the fog node, f_i could not surpass the available bandwidth of the fog node. Equation 4 indicates that the sum of the memory requirement of a set of application modules on the fog node, f_i is not more than the fog node's memory capacity. Finally, x_{ij} is a binary decision variable equal to 1 if the application module, a_j is allocated to f_i, and 0 otherwise. It ensures that an application is not assigned to multiple fog nodes.

4. Proposed Migration Management Algorithm

Providing support for applications demanding mobility is crucial for Fog environments. The users' mobility makes it necessary to move application modules from one fog node to another. Migration of applications reduces the delay, consequently ensuring that the applications' delay requirements are satisfied. However, application migration imposes unnecessary resource consumption; the migration count should be reduced. In migration, triggering the migration event and selecting the appropriate fog node is critical. Unnecessary activating of migration and wrong selection will lead to network overhead. The network delay between the fog node and the user is used to calculate the network overhead.

Algorithm 1: Migration Management Algorithm

Input: Mobility data of the user

Output: Destination fog node

1. **if** Delay > D_t **then** // D_t denotes delay threshold
2. Populate FNList[] with fog nodes present along the user's current direction
3. Find the distance, $D_{ij,}$ between the current location of the user and each element of FNList[]
4. Update FNList[] with fog nodes having D_{ij} < coverage radius
5. **for all** FNList[] **do**
6. Find connection time, *Ct*
7. Find execution time, *Et*
8. *Pt = Ct - Et*
9. **end for**
10. Sort FNList[] by Pt in descending order
11. **for all** FNList[] **do**
12. **if** $R_{req} < R_{cap} - R_{alloc}$ then // fog node has sufficient resources available
13. Select the fog node as the destination fog node
14. break
15. **else**
16. continue // for the rest of the fog nodes
17. **end if**
18. **end for**
19. Compute migration point, M_p based on user speed, application size, and network bandwidth // Algorithm 2
20. **if** user at migration point M_p **then**
21. Start migration process
22. Allocate resources at the destination fog node
23. Deallocate resources from the source fog node
24. **else**
25. No migration
26. **end if**
27. **end if**

Algorithm 2: Compute Migration Point

Input: user speed, dump size, bandwidth between fog nodes
Output: Migration point

1. Calculate migration time (*Tm*) by dividing dump size by bandwidth between fog nodes.
2. Calculate distance required to complete migration (*Dm*) by multiplying migration time and user speed.
3. Calculate migration point (*Mp*) by subtracting *Dm* from coverage radius.
4. Calculate distance (*Du*) between user's current position and its access point.
5. **if** *Du >= Mp* **then //** user approaches migration point
6. Start migration
7. **else**
8. No migration
9. **end if**

Since each migration event in the proposed work is triggered only when the network latency exceeds a predefined threshold, the likelihood of such overheads is relatively low. This event takes place when the IoT user moves far away from its serving node. The source fog node runs the destination node selection algorithm amongst the candidate fog nodes. By maximizing the function, the approach determines the optimal mapping of the application modules and the target fog node. This approach guarantees service continuity and QoS.The proposed algorithm takes user mobility data as its input. Whenever the source fog node (FN) finds that the user is moving towards the coverage boundary of its currently connected FN and is expected to leave, the migration decision process is initiated. It populates the FNList[] containing the list of FNs along the user's current direction. The distance between the user and the list of populated FNs is calculated. Accordingly, the FNList[] is updated to keep the FNs currently covering the IoT device. The connection time, *Ct*, and execution time, *Et*; for all the FNs in the list are calculated. A variable, *Pt*, is defined to store the difference value of *Ct* and *Et*. The algorithm aims to find a suitable fog node with a maximum *Pt* value. The proposal sorts the candidate FNs by *Pt* in descending order to check the resource availability. Thus, the set of FNs is sequentially checked for the availability of required resources, and if the resources are available, the destination node is chosen. The migration point, $M_{p,}$ is computed based on the mobility information of the user containing user speed, application size, and network bandwidth. The computation of M_p is explained in

algorithm 2. As soon as the user reaches the migration point, the source fog node initiates the migration process, and the resource allocation and deallocate process is started. It is essential to note that selecting the destination FN with this policy reduces the number of possible migrations in the system, improving the service continuity and quality of service. Algorithm 2 summarizes the computation of the migration point. Dynamic migration point considers the user's speed, dump size being migrated, and the bandwidth between fog nodes. The dynamic migration point, *Mp*, also considers the coverage radius of the connected access point. The process is initiated if the user has approached the computed migration point.

5. Simulation Results and Discussion

The simulation results are discussed here to assess the effectiveness of the proposed algorithm. The simulation assumes that processing will occur in fog devices and, if necessary, on the Cloud. All real-time processing will be done in fog environments using fog resources. Fog nodes have heterogeneous computing resources.MobFogSim toolkit, an extension of iFogSim, has been used to simulate the proposed algorithm. It is useful for modeling real-world mobile applications. Distinct features of MobFogSim facilitate different aspects of user mobility and container migration. Resource management policies are implemented using this simulator. It contains a specific resource management module that manages all resource allocation facets in fog and cloud environments. The input for users' mobility in the simulation is taken from the MobFogSim mobility dataset collected from the Luxembourg traffic [40].

5.1 Simulation Setup and Parameters

The algorithm is simulated over a 10 km x 10 km square region in which the coverage range of Fog nodes is assumed to be 1000 meters. The system consists of a cloud layer, a layer of fog nodes and a layer of IoT devices. The simulation parameters are outlined in Table 3. The system has a dense deployment of fog nodes. Fog nodes' processing power is randomly chosen from [1500-6000] MIPS.

Table 3. *Simulation parameters*

Parameter	**Value**
Map Scenario	
Scenario map size/ Area	10 km x 10 km
Access point coverage (radius)	1000 m
Number of fog devices	196
Density of fog devices per access point	1:1
Fog Device Characteristics	
Speed (MIPS)	1500 – 6000
RAM	8000 MB
Bandwidth	100 MBPS
Busy Power (MJ)	107.339
Idle Power (MJ)	83.433
IoT Device Characteristics	
Speed (MIPS)	500
RAM	1000 MB
Bandwidth	100 MBPS
Busy Power (MJ)	87.53
Idle Power (MJ)	82.44

The user's application size is taken as 128 MB. Several evaluation scenarios are carried out during the simulation. Initially, 100 IoT users' applications are submitted to the fog infrastructure in the evaluation scenario. Afterward, the number of users' applications increased progressively, reaching 400. Delay, downtime, migration time, network usage, number of migrations, and energy consumption are measured for this evaluation scenario. The round-trip time (RTT) and the throughput values among the fog nodes are based on real-life use cases. The migration is carried out using two distinct configurations of throughput and RTT values between fog nodes [11]. Each configuration indicates a particular network condition that may take place in reality. The throughput values and the associated RTT values within a fog environment are mentioned in Table 4. Configuration A corresponds to good network conditions based on fixed computers that are part of the local area network. In comparison, the other configuration represents poor network conditions that may exist between a smartphone connected to the Internet via 4G and a computer connected to the network via Ethernet. The simulations are run with these two network configurations.

Table 4. *Network configurations among fog nodes*

Configuration	**Throughput (Mbps)**	**RTT (ms)**
A	11.34	122.95
B	72.41	6.94

5.2 QoS Parameters

In the simulation, the following QoS parameters are used to assess the effectiveness of the algorithms:

5.2.1 Average Delay

Delay is the time the system needs to respond to a user's request after it has been sent. The delay depends on four basic parameters: transmission delay, execution delay, propagation delay, and queuing delay. The delay between the application, a_i running at the user's device and the fog device, f_j is calculated as follows:

$$D_{tot}^{ij} = D_{trans}^{ij} + D_{exe}^{ij} + D_{prop}^{ij} + D_{que}^{ij} \quad (6)$$

where D_{tot}^{ij} represents the total delay, D_{trans}^{ij} is the transmission delay, D_{exe}^{ij} is the execution delay, D_{prop}^{ij} is the propagation delay, and, D_{que}^{ij} is the queuing delay between the fog node, f_j and the user application a_i.

The transmission delay can be expressed as follows:

$$D_{trans}^{ij} = DU_{trans}^{ij} + DW_{trans}^{ij} \quad (7)$$

where DU_{trans}^{ij} represents the time taken to transmit the task generated by the application, a_i to the fog node f_j, and DW_{trans}^{ij} refers to the time needed to send the output of executed task to the user from the fog node. In the equation above, DU_{trans}^{ij} is referred to as the task size (T_i) divided by the transmission rate of the communication link, R_{ij} and it is expressed as:

$$DU_{trans}^{ij} = T_i / R_{ij} \quad (8)$$

where R_{ij} is calculated according to Shannon's capacity formula [41]. Given the channel bandwidth β and δ_{SINR} as signal-to-noise-plus-interference ratio as follows:

$$R_{ij} = \beta \times (1 + \delta_{SINR}) \quad (9)$$

On completion of the task processing at the Fog, the time consumed in sending back the result is calculated as:

$$DW_{trans}^{ij} = T_i' / R_{ij} \quad (10)$$

where T_i' is the result's size incurred from the fog node computation.

The time required for executing the user request is called the execution delay. Execution delay for the task *k* at the fog node is calculated as:

$$D_{exe}^{ij} = I_m / C_j^{max} \quad (11)$$

where I_m indicates the task's number of instructions in terms of MI, and C_j^{max} represents the computing power of the fog node f_j.

Propagation delay is the time required to transfer a data packet via the medium from one point to another. Propagation delay for a task running on a fog node is determined as follows:

$$D_{prop}^{ij} = 2 \times D_{uf} / P_s \quad (12)$$

where D_{uf} represents the user's distance from the connected fog node, and P_s is the propagation speed of the network. Queuing delay, being negligible, may be ignored.

5.2.2 Total Migration Time

Migration time is the time required for transferring a live running container from one fog node to another. The total migration time is modeled using different components: local computations times and transfer times[11][42]. Local computation time is the result of premigration and post-migration-related events. The premigration event comprises the time needed for selecting the destination fog node, namely, premigration time (T_{pm}) and the time consumed in reserving resources (T_{rsv}) at the chosen destination. The postmigration component comprises of commitment stage and activation stage. During these stages, the migration process is committed (T_{comm})and the migrated container service is resumed at the destination node (T_{rst}). The other component, transfer time, transfers a specific dump data, D_s and is network dependent. It combines two elements, migration transfer time, T_{tr}, and migration latency, T_{lat}, between the source and destination fog nodes. The computation time is machine dependent and is considered a constant, C_{mig}. The equation of migration time, T_{mig}, is expressed as:

$$T_{mig} = T_{tr} + T_{lat} + C_{mig} \quad (13)$$

where $C_{mig} = T_{pm} + T_{rsv} + T_{comm} + T_{rst}$

The migration transfer time is formulated as follows:

$$T_{tr} = D_s / R_{ij} \quad (14)$$

5.2.3 Total Downtime

Downtime is the time interval during which the application is stopped to perform the migration, and the user cannot access the service. It includes transferring the remaining memory dump and states and resuming the application on the destination fog node [11][42]. In downtime, a specific amount of memory dump (D_s') must be transferred, which is the final copy operation from the source node to the destination node. It also includes machine-dependent commitment time (T_{comm}) and container restoration time (T_{rst}).

$$T_{dt} = T_{tr}' + T_{lat}' + C_{dt} \quad (15)$$

where the machine dependent constant part, $C_{dt} = T_{comm} + T_{rst}$ and transmission time, $T_{tr}' = D_s' / R_{ij}$.

5.2.4 Total Network Usage

Network usage is the total data sent and received during the migration process.

$$NW_{tot} = D_{tot} \times T_{lat} \quad (16)$$

where D_{tot} is the size of data sent during the migration of the application modules and T_{lat} is the network delay between source and destination fog nodes. Total network usage is determined by summing the network consumtion incurred during each migration event.

5.2.5 Total Energy Consumption

Total energy consumption is the combination of two components: the energy consumed to transmit the task to the fog node and the energy consumed to execute the task. Total energy consumption can be written as follows:

$$E_{tot}^{ij} = E_{trans}^{ij} + E_{exe}^{ij} \quad (17)$$

where E_{trans}^{ij} is the energy consumed during transmission and E_{exe}^{ij} is the energy consumed during execution performed by the fog nodes in the system.

$$E_{trans}^{ij} = T_{trans} \times \lambda \quad (18)$$

where T_{trans} is transmission time, and λ is a constant related to the wireless interface [43].

$$E_{exe}^{ij} = T_{exe} \times \mu \quad (19)$$

where T_{exe} is the execution time, and μ is a coefficient denoting the energy consumption per CPU cycle.

5.3 Results

The proposed algorithm is compared with the algorithm based on the lowest latency-based strategy [5] to demonstrate its performance. The lowest latency-based strategy is an application migration algorithm that efficiently utilizes the various Fog node resources and chooses the appropriate Fog node for application migration. The algorithm selects the Fog node with the lowest end-to-end latency out of all the Fog nodes available with sufficient resources. For both algorithms, the simulation results are anlyzed based on the number of migrations, delay, downtime, migration time, network usage, and energy consumption.

5.3.1 Total Number of Migrations

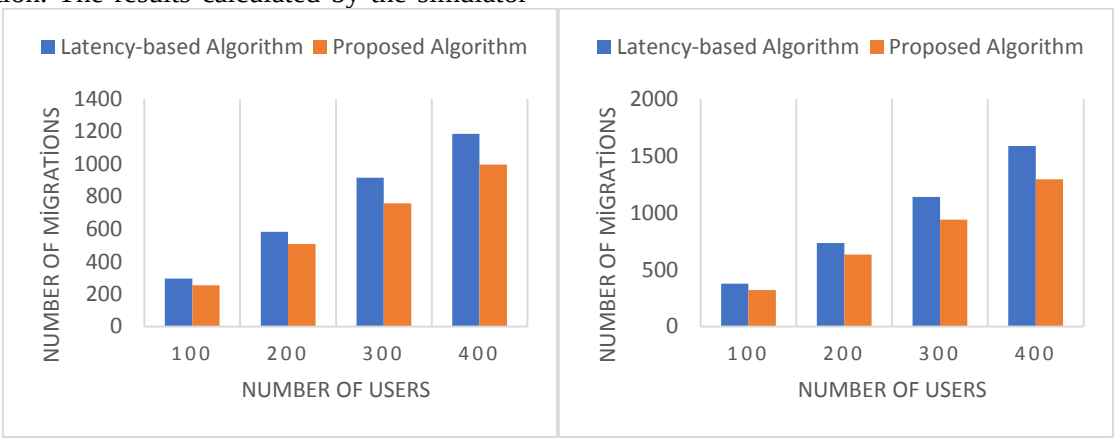
The number of migrations is the migration frequency that a user of IoT device experiences along its path. Although necessary, fewer application migrations should be made because each migration event consumes additional resources. The source fog nodes have information about the real-time mobility of departing devices (e.g., their direction and speed when within the current fog node's range). So, the connection and execution time of all the candidate fog nodes with IoT devices can be estimated for the migration decision process. As per the proposed algorithm, the number of possible migrations decreases by migrating the application modules to the appropriate node. The existing policies aim to lower the cost of migration by migrating application modules to new fog nodes without considering the current mobility information of devices, connection duration, and execution time.The analysis of the required migrations shows that the proposed approach lowers the number of migrations because it considers IoT devices' current mobility information, such as speed and direction. A reduction in the number of migrations results in less downtime for the user's application.Figure 2 shows the number of migrations for configurations A and B. Under both configurations, the proposed approach decreases the number of migrations to almost 18% compared to the existing approach. The reduction in migrations indicates that the locations where applications are placed are better suited for the user.

5.3.2 Average Delay

After the user has moved out of the range of the fog node currently hosting the user application, a higher delay may be experienced in the response received from the source fog node hosting the application. It is caused due to the increased count of hops between the user and the application hosted at the source fog node. Delay is the primary QoS metric evaluated in the proposed migration approach. In contexts where high performance is required in real-time, it is the factor that must be decreased.In the simulation, it is expected that there won't be any instances in which there are deficient fog resources to execute the services; thus, queuing delay is considered zero. Execution delay is the major component affecting the end-to-end delay for the user applications.The proposed approach outperforms the latency-based approach by choosing the most suitable destination fog node based on various parameters. A comparison of the analyzed scenarios is shown in Figure 3. The average delay is shown for different numbers of users in the simulation. The results in the figures are apparent indicators that implementing the proposed migration strategy would result in higher QoS for the end users. The proposed algorithm shows a reduction of up to 20% under poor network conditions and up to 17% under good network conditions when used to address the issue of application migration.

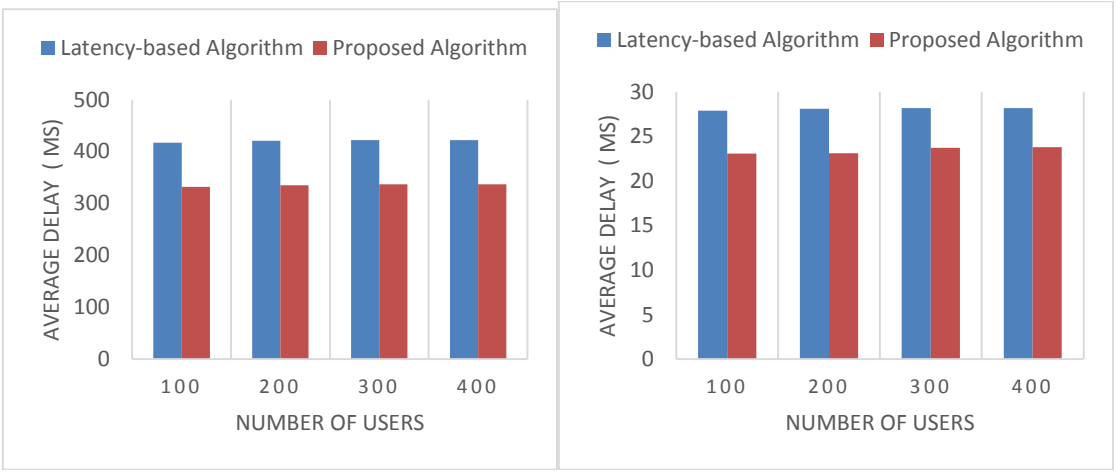
5.3.3 Total Downtime

The time during which service is not available should be minimized. The proposed approach reduces the unnecessary triggering of migration, which decreases the total downtime during the simulation. The results calculated by the simulator concerning downtime are given in Figure 4. As the number of users increases, the downtime improves further. On average, the proposed approach results in 13% and 20% less downtime than the existing approach.



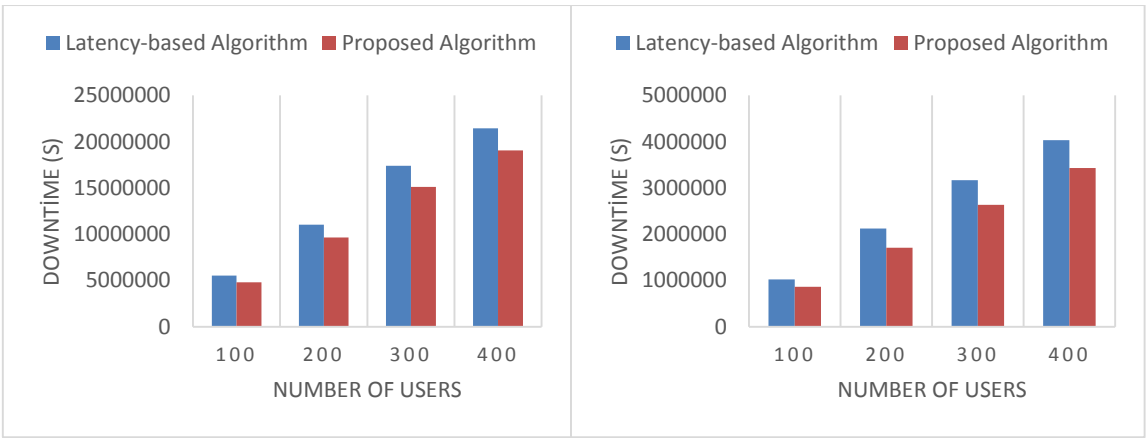
(a) Network Configuration A *(b) Network Configuration B*

Figure 2. *Total number of migrations under different network configurations*



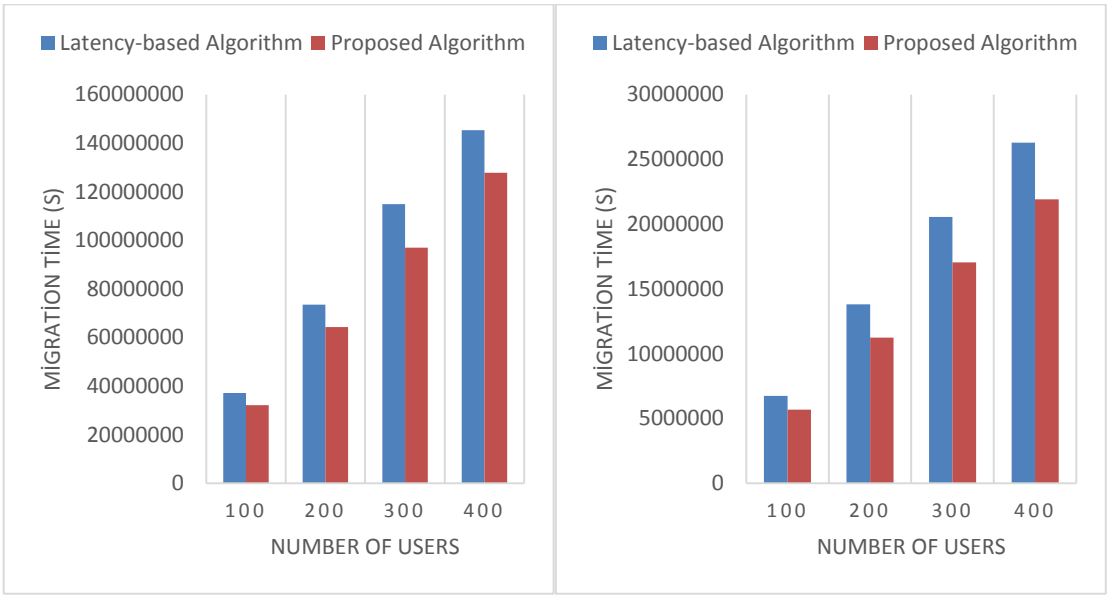
(a) Network Configuration A *(b) Network Configuration B*

Figure 3. *Average delay under different network configurations*



(a) Network Configuration A *(b) Network Configuration B*

Figure 4. *Total downtime under different network configurations*



(a) Network Configuration A *(b) Network Configuration B*

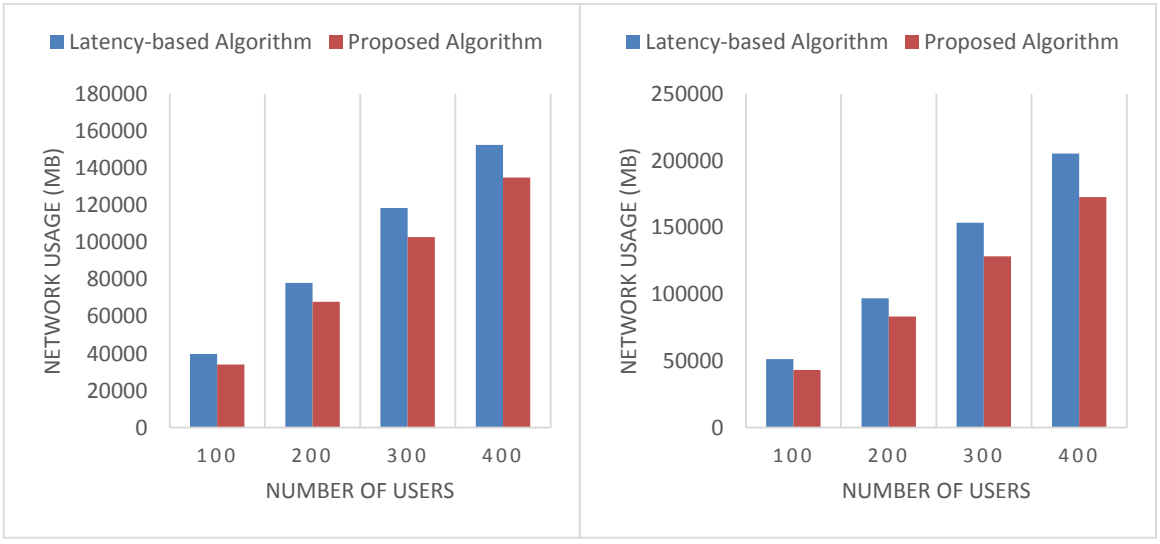
Figure 5. *Total migration time under different network configurations*

5.3.4 Total Migration Time

The time needed to transfer the application to the destination fog node should be minimized. The total migration time of the proposed technique is evaluated in comparison to the existing technique. The total migration time is shown in Figure 5. Simulation with the proposed approach presents a migration time of up to 13% shorter under configuration A. In comparison, under configuration B, the migration time is reduced by up to 19% than the existing policy.The proposed approach reduces the total number of migrations and, thus, lessens the total migration time compared to the latency-based method, which has a higher number of total migrations.

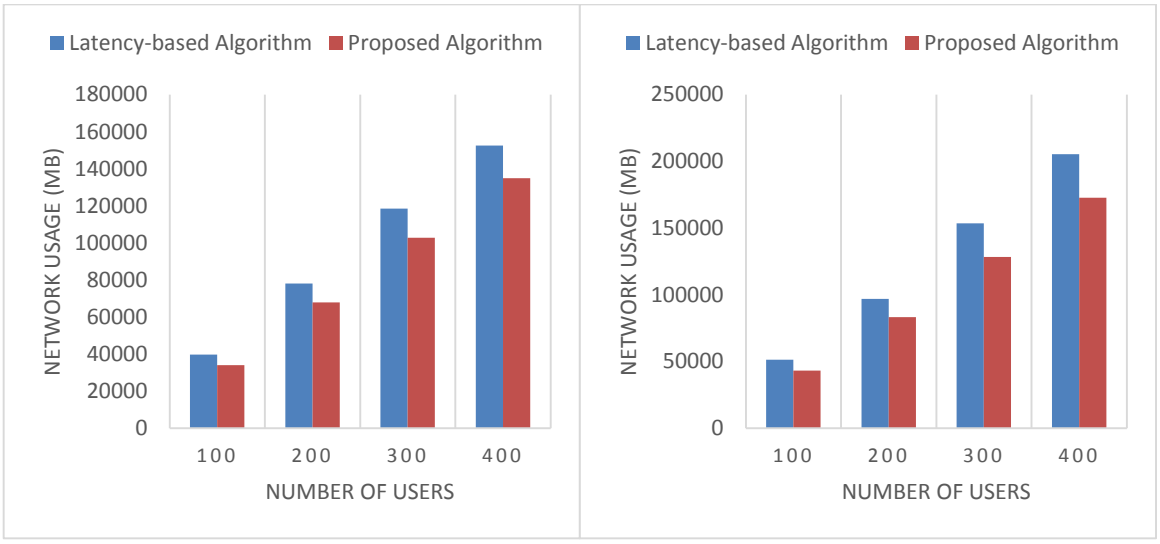
5.3.5 Total Network Usage

Uncontrolled network use may cause congestion, declining the application's performance. The network usage during application migration for different simulating approaches is portrayed in Figure 6. The proposed algorithm works slightly better in this situation since it decreases the number of migrations of application modules and thus decreases their network usage. As the number of migrating events decreases, network usage decreases. The proposed algorithm reduces network usage by up to 16%.



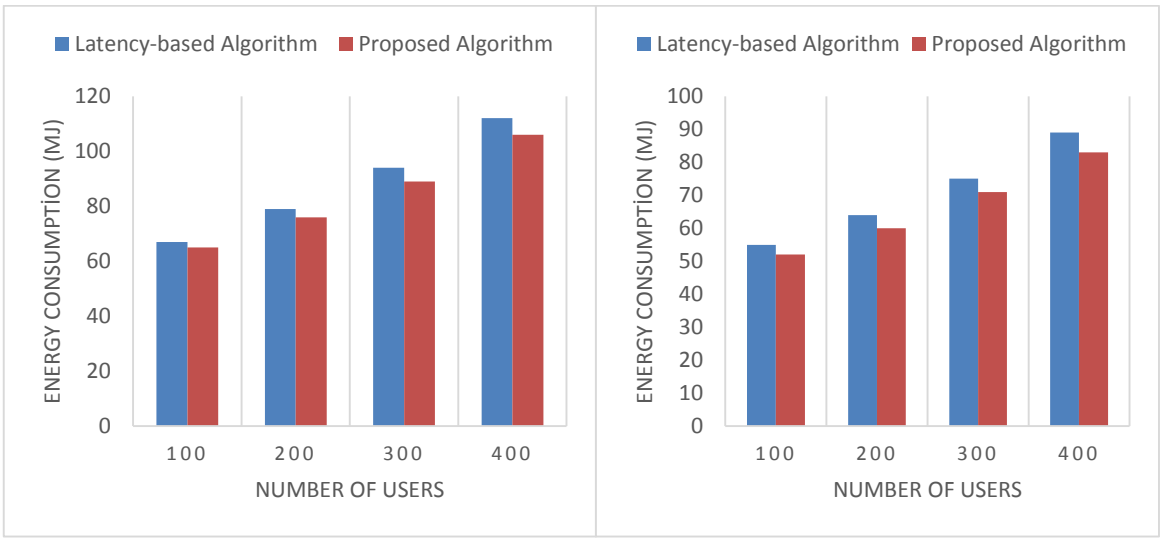
(a) Network Configuration A *(b) Network Configuration B*

Figure 6. *Total network usage under different network configurations*



(a) Network Configuration A *(b) Network Configuration B*

Figure 6. *Total network usage under different network configurations*



(a) Network Configuration A *(b) Network Configuration B*

Figure 7. *Total energy consumption under different network configurations*

5.3.6 Total Energy Consumption

The system's overall energy consumption rises when more IoT users are added. Another aspect of the system's energy usage is the frequency of migration.The lesser the system's migration frequency, the lesser the energy consumption. If the fog nodes are battery-operated, the lower energy consumption may enable a longer node life. The energy consumption during module migration for both policies is presented in Figure 7. The proposed migration approach performs well in handling energy usage by decreasing the migration frequency. The suggested algorithm is perceived to consume less energy by up to 7% compared to the existing latency-based algorithm.

5.3.7 Qualitative Analysis

This section analyzes qualitative parameters inferred from this study and relevant to migration management. Different parameters considered are connectivity, availability, performance, and resource utilization [44].Connectivity: It is a state during which a valid connection exists between various devices in the system to execute applications. IoT device users move out of the associated access points' coverage area, breaking the connection between the user and the fog resource. The connectivity failure will result in service disruption affecting the QoS. Maintaining connectivity without service disruption is essential while migrating to a suitable destination node. The connectivity parameter is dependent on the number of migrations in the system. Each migration event contributes to

connectivity. Appropriate selection of the destination node in the proposed approach reduces the number of migrations and improves connectivity.Availability: The availability of fog service is a way to specify the system's capability to ensure that the requested resources are available with the expected performance to service user task requests. Downtime and migration time determine the availability of the services in the fog environments. In fog computing, the mobility of the end user causes service migration which in turn causes availability issues in the fog environments. The proposed algorithm reduces the downtime and migration time parameters and improves availability. Performance: Lowering application delay is important to achieving effective application performance. As the user leaves the coverage area of its associated fog node, the network delay increases as more hops are needed to communicate with the serving fog node. The proposed system has lowered the average delay perceived by the end users.Resource Utilization: Resource utilization is a performance metric that gives feedback on how efficiently various resources are allocated to the application modules. Appropriate migration decisions improve the utilization of resources. Compared to the existing technique, the proposed technique makes resource utilization more efficient by reducing network usage and energy consumption. It helps in avoiding congestion in the network. Improvement in energy efficiency may consequently allow the battery-operated fog nodes to have longer battery life.

6. Conclusions and Future Work

The exponential growth of IoT devices makes it challenging to maintain QoS in fog computing environments. The migration is triggered whenever a user of an IoT device begins to move from one service area to another, and the associated application needs to be migrated to a suitable fog node. The MobFogSim tool has been used to simulate application migration. The simulation tool takes account of the wireless connectivity, the user's mobility and the application migration process. The migration decision has been proposed considering various parameters, such as connection time between users and fog nodes and the application execution time. Each network parameter has been mathematically modeled, considering the highly mobile network. Simulations have been done with two network configurations corresponding to good and poor network conditions.The comparison analysis demonstrates that the proposed migration approach significantly improves various parameters. Under both network configurations, the proposed approach decreases the number of migrations to almost 18%. A decrease of up to 20% in average delay under poor network conditions and up to 17% under good network conditions is achieved. The proposed migration approach produces a downtime 20 % lower than the existing approach under both conditions. The migration time of up to 19% shorter is obtained under good network conditions, while, under poor network conditions, the migration time is reduced by up to 13% than the existing policy. The proposed algorithm decreases network usage by up to 16% and consumes less energy by up to 7% compared to the existing algorithm. The simulation results show that the proposed algorithm enhances different QoS parameters significantly. Further, qualitative parameters also indicate the improvement in the experience perceived by the users. Low latencies, low network usage and improved energy efficiency benefit the users of the proposed approach. The proposed technique also keeps Fog nodes operational for a longer duration if these are battery-operated. In contrast to the current situation, where only IoT devices are considered mobile, choosing an appropriate node will be more difficult when fog nodes are also mobile. As part of future work, this work may be extended to investigate the mobility of fog nodes and its effect on the QoS and the performance of the fog environment.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1.] Afrin, M., Jin, J., Rahman, A., Gasparri, A., Tian, Y.-C., & Kulkarni, A. (2021). Robotic edge

resource allocation for agricultural cyber-physical system. *IEEE Transactions on Network Science and Engineering.* https://doi.org/10.1109/TNSE.2021.3103602

[2.] Alam, M., Ahmed, N., Matam, R., & Barbhuiya, F. A. (2022). L3Fog: Fog node selection and task offloading framework for mobile IoT. In *INFOCOM Workshops 2022 – IEEE Conference on Computer Communications Workshops*. https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798118

[3.] Anawar, M. R., Wang, S., Azam Zia, M., Jadoon, A. K., Akram, U., & Raza, S. (2018). Fog computing: An overview of big IoT data analytics. *Wireless Communications and Mobile Computing, 2018*, Article 7157192. https://doi.org/10.1155/2018/7157192

[4.] Bellavista, P., & Zanni, A. (2017). Feasibility of fog computing deployment based on Docker containerization over Raspberry Pi. In *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3007748.3007777

[5.] Bi, Y., Han, G., Lin, C., Deng, Q., Guo, L., & Li, F. (2018). Mobility support for fog computing: An SDN approach. *IEEE Communications Magazine, 56*(5), 53–59. https://doi.org/10.1109/MCOM.2018.1700908

[6.] Bittencourt, L. F., Diaz-Montes, I. J., Buyya, R., Rana, O. F., & Parashar, M. (2017). Mobility-aware application scheduling in fog computing. *IEEE*.

[7.] Bonomi, F., Milito, R., Natarajan, P., & Zhu, J. (2014). Fog computing: A platform for Internet of Things and analytics. In *Studies in Computational Intelligence* (Vol. 546). https://doi.org/10.1007/978-3-319-05029-4_7

[8.] Clark, C., et al. (2005). Live migration of virtual machines. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation (NSDI'05)* (pp. 273–286). USENIX Association.

[9.] Codeca, L., Frank, R., & Engel, T. (2015). Luxembourg SUMO Traffic (LuST) scenario: 24 hours of mobility for vehicular networking research. In *2015 IEEE Vehicular Networking Conference (VNC)* (pp. 1–8). https://doi.org/10.1109/VNC.2015.7385539

[10.] Cisco Public. (2015). *Fog computing and the Internet of Things: Extend the cloud to where the things are*. https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[11.] Deswal, S., & Singhrova, A. (2020). Quality of service provisioning using multicriteria handover strategy in overlaid networks. *Malaysian Journal of Computer Science, 33*(1), 1–21. https://doi.org/10.22452/MJCS.VOL33NO1.1

[12.] Ghanavati, S., Abawajy, J., & Izadi, D. (2022). An energy aware task scheduling model using ant-mating optimization in fog computing environment. *IEEE Transactions on Services Computing, 15*(4), 2007–2017. https://doi.org/10.1109/TSC.2020.3028575

[13.] Goudarzi, M., Palaniswami, M., & Buyya, R. (2021). A distributed application placement and migration management techniques for edge and fog computing environments. *arXiv*. http://arxiv.org/abs/2108.02328

[14.] Guerrero, C., Lera, I., & Juiz, C. (2019). A lightweight decentralized service placement policy for performance optimization in fog computing. *Journal of Ambient Intelligence and Humanized Computing, 10*(6), 2435–2452. https://doi.org/10.1007/s12652-018-0914-0

[15.] Gupta, H., Dastjerdi, A. V., Ghosh, S. K., & Buyya, R. (2017). 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, 47*(9), 1275–1296. https://doi.org/10.1002/spe.2509

[16.] Islam, M., Razzaque, A., & Islam, J. (2016). A genetic algorithm for virtual machine migration in heterogeneous mobile cloud computing. In *Proceedings of the 2016 International Conference on Networking Systems and Security (NSysS)*. https://doi.org/10.1109/NSysS.2016.7400696

[17.] Kaur, K., Dhand, T., Kumar, N., & Zeadally, S. (2017). Container-as-a-service at the edge: Trade-off between energy efficiency and service availability at fog nano data centers. *IEEE Wireless Communications, 24*(3). https://doi.org/10.1109/MWC.2017.1600427

[18.] Liao, S., Li, J., Wu, J., Yang, W., & Guan, Z. (2019). Fog-enabled vehicle as a service for computing geographical migration in smart cities. *IEEE Access, 7*, 2890298. https://doi.org/10.1109/ACCESS.2018.2890298

[19.] Liu, C., Wang, J., Zhou, L., & Rezaeipanah, A. (2022). Solving the multi-objective problem of IoT service placement in fog computing using Cuckoo search algorithm. *Neural Processing Letters, 54*(3), 1823–1854. https://doi.org/10.1007/s11063-021-10708-2

[20.] Lopes, M. M., Higashino, W. A., Capretz, M. A. M., & Bittencourt, L. F. (2017). MyiFogSim. In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing (UCC '17 Companion)* (pp. 47–52). https://doi.org/10.1145/3147234.3148101

[21.] Machen, A., Wang, S., Leung, K. K., Ko, B. J., & Salonidis, T. (2018). Live service migration in mobile edge clouds. *IEEE Wireless Communications, 25*(1), 140–147. https://doi.org/10.1109/MWC.2017.1700011

[22.] Mahmud, R., Pallewatta, S., Goudarzi, M., & Buyya, R. (2022). iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software, 190*, 111351. https://doi.org/10.1016/j.jss.2022.111351

[23.] Mahmud, R., Ramamohanarao, K., & Buyya, R. (2018). Latency-aware application module management for fog computing environments. *ACM Transactions on Internet Technology, 19*(1), 1–21. https://doi.org/10.1145/3186592

[24.] Mahmud, R., Srirama, S. N., Ramamohanarao, K., & Buyya, R. (2019). QoE-aware placement of applications in fog computing environments. *Journal of Parallel and Distributed Computing, 132*, 62–71. https://doi.org/10.1016/j.jpdc.2018.03.004

[25.] Martin, J. P., Kandasamy, A., & Chandrasekaran, K. (2018). Exploring the support for high performance applications in the container runtime environment. *Human-centric Computing and Information Sciences, 8*(1). https://doi.org/10.1186/s13673-017-0124-3

[26.] Mishra, M., Roy, S. K., Mukherjee, A., De, D., Ghosh, S. K., & Buyya, R. (2020). An energy-aware multi-sensor geo-fog paradigm for mission critical applications. *Journal of Ambient Intelligence and Humanized Computing, 11*(8). https://doi.org/10.1007/s12652-019-01481-1

[27.] Naha, R. K., & Garg, S. (2021). Multi-criteria-based dynamic user behaviour-aware resource allocation in fog computing. *ACM Transactions on Internet of Things, 2*(1), 1–31. https://doi.org/10.1145/3423332

[28.] Naha, R. K., et al. (2018). Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access, 6*, 47980–48009. https://doi.org/10.1109/ACCESS.2018.2866491

[29.] Paul, J., Kandasamy, M. A., & Martin, J. P. (2020). Mobility aware autonomic approach for migration of application modules in fog computing environment. *Journal of Ambient Intelligence and Humanized Computing.* https://doi.org/10.1007/s12652-020-01854-x

[30.] Puliafito, C., Vallati, C., Mingozzi, E., Merlino, G., Longo, F., & Puliafito, A. (2019). Container migration in the fog: A performance evaluation. *Sensors, 19*(7), 1488. https://doi.org/10.3390/s19071488

[31.] Puliafito, C., et al. (2019). MobFogSim: Simulation of mobility and migration for fog computing. *Simulation Modelling Practice and Theory, 94*, 102062. https://doi.org/10.1016/j.simpat.2019.102062

[32.] Sarrafzade, N., Entezari-Maleki, R., & Sousa, L. (2022). A genetic-based approach for service placement in fog computing. *Journal of Supercomputing, 78*(8), 10854–10875. https://doi.org/10.1007/s11227-021-04254-w

[33.] Satyanarayanan, M. (2017). The emergence of edge computing. *Computer, 50*(1). https://doi.org/10.1109/MC.2017.9

[34.] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27*(3), 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

[35.] Shckhar, S., Chhokra, A., Sun, H., Gokhale, A., Dubey, A., & Koutsoukos, X. (2019). URMILA: A performance and mobility-aware fog/edge resource management middleware. In *Proceedings of the 2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*. https://doi.org/10.1109/ISORC.2019.00033

[36.] Shi, W., & Dustdar, S. (2016). The promise of edge computing. *Computer, 49*(5). https://doi.org/10.1109/MC.2016.145

[37.] Statista. Vailshery, L. S. (2021). Number of IoT connected devices worldwide 2019–2021, with forecasts to 2030. https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[38.] Wang, D., Liu, Z., Wang, X., & Lan, Y. (2019). Mobility-aware task offloading and migration schemes in fog computing networks. *IEEE Access, 7*, 43356–43368. https://doi.org/10.1109/ACCESS.2019.2908263

[39.] Wang, S., Urgaonkar, R., He, T., Chan, K., Zafer, M., & Leung, K. K. (2017). Dynamic service placement for mobile micro-clouds with predicted future costs. *IEEE Transactions on Parallel and Distributed Systems, 28*(4). https://doi.org/10.1109/TPDS.2016.2604814

[40.] Wang, S., Guo, Y., Zhang, N., Yang, P., Zhou, A., & Shen, X. (2021). Delay-aware microservice coordination in mobile edge computing: A reinforcement learning approach. *IEEE Transactions on Mobile Computing, 20*(3). https://doi.org/10.1109/TMC.2019.2957804

[41.] Yi, S., Li, C., & Li, Q. (2015). A survey of fog computing: Concepts, applications, and issues. In *MOBIDATA 2015: Workshop on Mobile Big Data*. https://doi.org/10.1145/2757384.2757397

[42.] Zhao, D., Zou, Q., & Boshkani Zadeh, M. (2022). A QoS-aware IoT service placement mechanism in fog computing based on open-source development model. *Journal of Grid Computing, 20*(2). https://doi.org/10.1007/s10723-022-09604-3

[43.] Zhao, X., Liu, J., Ji, B., & Wang, L. (2021). Service migration policy optimization considering user mobility for e-healthcare applications. *Journal of Healthcare Engineering, 2021*, Article 9922876. https://doi.org/10.1155/2021/9922876

[44.] Zhu, C., Pastor, G., Xiao, Y., Li, Y., & Ylä-Jääski, A. (2018). Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In *2018 IEEE SECON*. https://doi.org/10.1109/SAHCN.2018.8397129