digital

Article

On the Effectiveness of Fog Offloading in a Mobility-Aware Healthcare Environment

Ferdous Sharifi [1,2], Ali Rasaii [2], Amirmohammad Pasdar [1,3], Shaahin Hessabi [2] and Young Choon Lee [1,*]

[1] School of Computing, Macquarie University, Sydney 2109, Australia; ferdous.sharifi@hdr.mq.edu.au or f.sharifi95@sharif.edu (F.S.); amirmohammad.pasdar@hdr.mq.edu.au or a.pasdar@adfa.edu.au (A.P.)
[2] Department of Computer Engineering, Sharif University of Technology, Tehran 11155-9517, Iran; arasaii@mpi-inf.mpg.de (A.R.); hessabi@sharif.edu (S.H.)
[3] School of Systems & Computing, University of New South Wales, Canberra, ACT 2600, Australia
[*] Correspondence: young.lee@mq.edu.au

Abstract: The emergence of fog computing has significantly enhanced real-time data processing by bringing computation resources closer to data sources. This adoption is very beneficial in the healthcare sector, where abundant time-sensitive processing tasks exist. Although such adoption is very promising, there is a challenge with the limited computational capacity of fog nodes. This challenge becomes even more critical when mobile IoT nodes enter the network, potentially increasing the network load. To address this challenge, this paper presents a framework that leverages a Many-to-One offloading (M2One) policy designed for modelling the dynamic nature and time-critical aspect of processing tasks in the healthcare domain. The framework benefits the multi-tier structure of the fog layer, making efficient use of the computing capacity of mobile fog nodes to enhance the overall computing capability of the fog network. Moreover, this framework accounts for mobile IoT nodes that generate an unpredictable volume of tasks at unpredictable intervals. Under the proposed policy, a first-tier fog node, called the coordinator fog node, efficiently manages all requests offloaded by the IoT nodes and allocates them to the fog nodes. It considers factors like the limited energy in the mobile nodes, the communication channel status, and low-latency demands to distribute requests among fog nodes and meet the stringent latency requirements of healthcare applications. Through extensive simulations in a healthcare scenario, the policy's effectiveness showed an improvement of approximately 30% in average delay compared to cloud computing and a significant reduction in network usage.

Keywords: fog computing; Internet of Things; computation offloading; fog offloading; mobility-aware offloading; healthcare-monitoring system

check for updates

Citation: Sharifi, F.; Rasaii, A.; Pasdar, A.; Hessabi, S.; Lee, Y.C. On the Effectiveness of Fog Offloading in a Mobility-Aware Healthcare Environment. *Digital* **2023**, *3*, 300–318. https://doi.org/10.3390/digital3040019

Academic Editor: Geoffrey Fox

Received: 9 October 2023
Revised: 11 November 2023
Accepted: 20 November 2023
Published: 23 November 2023

1. Introduction

Today, a tremendous amount of data is generated from massively distributed smart devices that comprise the Internet of Things (IoT). The International Data Corporation (IDC) has projected that the digital data volume will surge from the 33 zettabytes in 2018 to 175 zettabytes by 2025 [1]. In contemporary implementations of IoT applications, most data requiring storage, analysis, and decision-making are typically transmitted to cloud-based data centres [2]. This process places significant strain on the network infrastructure, especially in terms of data transmission costs. A more-critical challenge arises for time-sensitive and location-aware applications, such as patient monitoring or autonomous vehicles [3]. Given these applications' stringent demands for ultra-low latency, relying on distant cloud resources becomes inadequate. Additionally, most IoT devices have limited power resources, and to prolong their lifespan, it is crucial to manage power consumption by scheduling computational tasks on devices with higher power and computational capabilities [4].

In 2014, CISCO introduced fog computing as a solution to the aforementioned challenges [5]. Fog computing strategically situates computation, storage, networking, decision-making, and data management between IoT devices and the cloud. The research community has also proposed other computing paradigms, like edge computing, to tackle these issues [3]. Although some papers use fog computing and edge computing interchangeably, they are not synonymous. The OpenFog Consortium draws a clear distinction, describing fog computing as a hierarchical approach that delivers computing, networking, storage, and decision-making capabilities across the entire spectrum from the cloud to IoT devices [6]. In contrast, edge computing primarily operates near IoT devices, constituting the initial hop from these devices rather than encompassing the entirety of IoT nodes [3].

Figure 1 presents the fog computing framework used in this paper. This framework comprises three layers: the IoT layer, housing end devices; the fog layer, featuring fog nodes arranged hierarchically with limited computing power; and the cloud layer, where cloud data centres are situated.

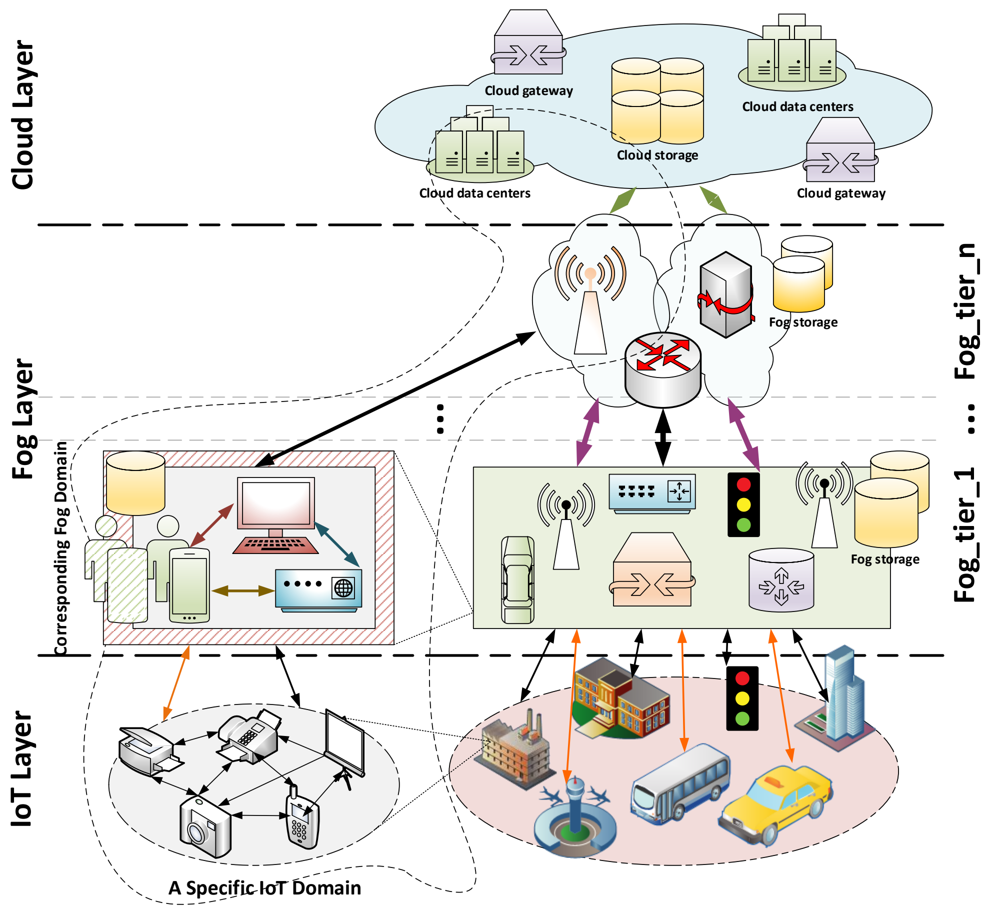


Figure 1. System model.

The IoT layer is where IoT devices, or end devices, are located and raw data are produced. IoT nodes are diverse regarding computing power, storage capability, and the communication protocol. The fog layer consists of all nodes located in the thing-to-cloud continuum. These nodes are divided into several tiers. Fog nodes, which are located in the vicinity of IoT nodes, are on the first tier. These nodes are typically one hop away from IoT nodes. The nodes that are two hops away are on the second tier, and so on. The last tier, which is the closest tier to the cloud, is called tier n. As we move from things to the cloud, the computing power, storage capability, and communication link bandwidth increase. Offloaded requests can be processed on the first tier of the fog layer or offloaded to higher tiers or in the cloud. Horizontal offloading is another feature of this framework. Horizontal offloading means sending a request to a node, not in the corresponding thing to the cloud continuum, but added to the path to take advantage of fog computing. Therefore, this framework considers fog-to-upper tier fog and fog-to-same tier fog interactions.

The cloud layer is where cloud servers with massive computing and storage resources are located. If fog nodes cannot fulfil the IoT application's objectives (for example, necessary computing capabilities and memory or deadline), requests are offloaded to the cloud. Cloud servers can perform heavy computational processing on the data received from fog nodes and permanently store information.

Although fog computing brings several advantages, some challenges should be addressed to reap all the benefits of fog computing. These challenges include architecture design, mobility management, content caching, and computation offloading [7]. Among these challenges, in this work, we focused on computation offloading. Computation offloading enables IoT devices to offload computation tasks to fog or cloud servers and receive the results after the servers execute the tasks [8]. In this way, we can free end devices from heavy computing tasks, reduce their energy consumption, and significantly advance the completion time of tasks. However, how many offloaded tasks should be executed in the fog layer and which fog nodes should execute them are vital problems that must be solved.

This paper presents an approach to the healthcare-monitoring environment, incorporating the mobility of patients and fog nodes. Our approach focuses on collecting vital signs from each patient and transmitting these data to a designated fog node, known as the coordinator fog node, for processing. By accounting for mobility and simulating real-world scenarios, we created a dynamic environment where task generation and fog node computing capacity evolve over time. To effectively manage the generated tasks and allocate them to the existing fog nodes in the area, we propose a many-to-one computation offloading policy called the M2One policy. Under this policy, all data collected by the sensors are forwarded to the coordinator fog node, which assumes responsibility for distributing these tasks among available fog nodes within the network. This distribution is performed to meet the stringent latency requirements of healthcare applications while considering the constraints of the available fog nodes. The coordinator fog node, conveniently situated just one hop away from the sensors, is the entry point to the cloud layer in traditional cloud computing.

The coordinator fog node's task allocation decisions hinge on several critical factors, including the computing power of fog nodes, the amount of time spent in the processing queue of fog nodes, which is known as the queuing delay, the communication delay, the communication channel status in terms of packet loss, and the energy constraints of the mobile fog nodes. These five factors are important because they directly affect latency, and ignoring any of them leads to a long execution time or request loss. A long execution time happens when a fog node receives a noisy request (a request with a low signal-to-noise ratio) due to an inappropriate communication channel. In this case, the coordinator fog node has to resend the offloaded request to another computing node. Request loss occurs when the offloaded request misses its deadline due to a long queuing delay or when a mobile fog node becomes unavailable due to limited energy.

The main contributions are as follows:

- Development of a many-to-one computation offloading policy: We introduce a novel approach specifically designed to manage multi-objective computation offloading, considering the dynamic mobility of fog nodes. This strategy employs advanced algorithms that dynamically allocate and distribute computational tasks in a many-to-one setup. It assesses queuing delays in fog nodes, the communication channel status, and the battery life of the mobile fog nodes. This method addresses the complexities presented by the mobile nature of fog nodes.
- Deploying of a multi-tier fog layer for enhanced resource utilisation: We adopted a multi-tier fog layer to maximise the utilisation of diverse fog node capacities. This unique setup involves intelligently allocating tasks across various tiers, leveraging their distinct computational capabilities to enhance overall network efficiency.

- Validation of the proposed policy in a dynamic healthcare environment: The effectiveness of the proposed policy was validated by implementing it within a healthcare-monitoring system. This involved a simulated healthcare scenario that accounted for the dynamic mobility of both fog nodes and patients. The created dynamic environment closely mimicked real-world conditions, where the fog layer's task generation and computing capabilities evolve over time.

This paper significantly extends our previous work [9] by (1) incorporating patient mobility into our model, creating a dynamic fog computing environment with varying data generation rates, and (2) conducting extensive experiments for the comprehensiveness of our findings.

Our simulations using the iFogSim simulator [10] revealed a request threshold for optimal fog capacity usage and highlighted the Quality of Service (QoS) benefits of increasing the computing capacity of the fog nodes. Additionally, our findings showed that the current fog nodes efficiently handle requests from mobile patients, but the same number of requests from static patients leads to higher average delays. Overall, our proposed method consistently outperformed traditional cloud computing, reducing average delays by an impressive 30%.

The rest of this paper is organised as follows. In Section 2, a review of the related work is presented. Then, we discuss the mobility model of fog nodes and present our computation offloading policy, the M2One policy, in Section 3. Section 4 describes the simulation settings. Our experimental results are discussed in Section 5. Finally, the paper is concluded in Section 6.

2. Related Work

In recent years, in the field of fog and edge computing, the research on computation offloading strategies has become a hot issue. Researchers have proposed various computation offloading strategies for different goals. These proposed strategies generally can be categorised into single-objective and multi-objective approaches.

In a single-objective approach, the computation-offloading strategy aims to decide where a task should be executed to improve the desired quality of service parameter, which can be latency, energy consumption, or cost. In contrast, in a multi-objective method, the focus is on the effect of several parameters on each other, and the decision about where the task should be executed is made based on the trade-off between several parameters.

2.1. Single-Objective Approaches

In the realm of vehicular networks, an approach aiming to strike a balance between latency and task quality allocation was introduced by [11]. To commence the process, a client vehicle initiates a one-hop probe message broadcast to identify nearby fog nodes within its communication range and, subsequently, collects their responses. Following this, the client vehicle transmits a request to a designated zone head, including details regarding the offloading tasks and potential fog node candidates. Ultimately, the zone head employs a task-allocation algorithm to make informed decisions about task execution locations. An effective resource provisioning strategy was introduced by [12] to mitigate latency violations. This approach leverages Bayesian learning techniques to make informed decisions regarding the dynamic scaling of fog resources, determining both resource increases and decreases as needed. A policy for on-demand computation offloading in fog networks was presented in [13]. In this approach, a host node seeking additional computational resources beyond its own capabilities dynamically identifies accessible nodes and establishes an on-demand local resource provider network. A framework for multi-hop computation offloading in vehicular fog computing was introduced by [14]. This framework deals with critical decisions such as choosing between in-vehicle or remote computation, selecting appropriate fog nodes, and distributing the load between inter-fog nodes, all guided by the queuing theory principles. In mobile edge computing, a decision-making system for offloading, driven by user context information, was introduced by [15]. This method encompasses four distinct phases: monitoring, analysis, planning, and execution, enabling it to determine whether to execute computations locally, offload them to edge devices, or send them to the cloud.

In [16], two distinct challenges related to optimal computation offloading were put forth: one involving energy constraints and the other centred around time constraints. In each of these problems, they maintained one parameter as a constant and aimed to minimise the other. Their approach involved the application of a straightforward greedy methodology to formulate a computation-offloading strategy. This strategy considered various factors, including the attributes of the communication channels, computation and communication power consumption models, the status of already assigned and allocated tasks, as well as the characteristics of the current task. A proactive task-offloading framework known as the virtual edge was introduced by [17]. This innovative framework leverages the concept of virtual edge nodes, each comprising multiple vehicles with characteristics such as low relative moving speeds and strong wireless connectivity. When a vehicle encounters the need to offload computation tasks, it creates a new virtual edge and transfers the tasks to it or delegates them to a nearby virtual edge, offering a solution to the task-offloading challenge within this context.

To extend the operational lifespan of battery-powered fog nodes, a novel smart energy management approach was proposed by [18]. Their solution involved equipping fog nodes with solar panels that facilitate battery recharging. These nodes effectively manage their energy resources by dynamically switching between requesting and computing devices as needed. The study conducted by [19] focused on developing a resource scheduling and a dynamic offloading optimisation model tailored for a multi-user mobile edge computing system. Through their work, they successfully formulated and solved an optimisation problem. This endeavour led to the creation of an iterative algorithm, which, in turn, enabled the determination of optimal strategies for various factors, including transmission power allocation, clock frequency control, offloading ratio, and the received power split ratio within the system.

2.2. Multi-Objective Approaches

The research conducted in [20] centred on addressing latency and reliability-sensitive computational tasks for fog computing in a swarm of drones application. They formulated an optimisation problem that concurrently considered reliability, latency, and energy consumption for task allocation. Their solution involved the development of a distributed task-allocation algorithm based on the proximal Jacobi ADMM method. To minimise the cost of energy consumption, delay, and price, Reference [21] proposed a user-centric computation offloading model. Their approach considered a mixed integer nonlinear programming model as an optimisation problem. They proposed a branch-and-bound algorithm that relied on linear relaxation improvement to solve it. In a different context, a dynamic computing offloading model based on genetic algorithms was introduced by [22] to reduce the energy consumption and the delay of task execution in edge computing networks. This model incorporated a task weight cost model that considered both processing delay and energy consumption.

In [23], the authors considered various factors, including energy consumption, transmission delay, QoS requirements, power limits, and wireless fronthaul constraints, in the context of fog-computing-based healthcare-monitoring systems. They formulated an optimisation problem aimed at minimising the cost–utility of medical users. This problem was transformed into three decoupling sub-problems for the solution. In the research conducted in [24], the authors delved into a joint problem involving cooperative resource assignment and computation task offloading in mobile edge computing. They aimed to minimise latency while adhering to transmission power, energy consumption, and CPU cycle frequency constraints. To address this, they proposed an iterative algorithm based on Lagrangian dual-decomposition, monotonic optimisation techniques, and the Shengjin formula method.

Although the mentioned prior works showcased certain advantages and improvements, the challenge lied in their comprehensive evaluation metrics. Most previous methodologies focused on limited objectives, such as latency or energy consumption, and some considered only two specific objectives. This limited focus on their evaluation criteria may impact the accuracy and completeness of their results. Moreover, as indicated in Table 1, the frameworks considered in these studies differed, potentially neglecting important features of a real fog computing environment (e.g., mobility of fog nodes and IoT nodes) and the full capabilities of a fog computing architecture (e.g., multi-tier fog layer, horizontal offloading). Therefore, our objective was to propose a multi-objective computation-offloading strategy that simultaneously considers latency, energy consumption, and communication channel status. Additionally, we aimed to leverage the multi-tier nature of fog computing in a simulation of a real-world environment to enhance the computing capacity of the fog layer.

Table 1. Comparison of the architectures used in the previous works.

Reference	[25]	[11]	[26]	[13]	[20]	[12]	M2One Policy
Three-Layer Architecture *	✓	×	✓	×	×	✓	✓
Multi-Tier Fog Layer	×	✓	×	×	×	✓	✓
Horizontal Offloading	✓	×	✓	✓	✓	✓	✓
IoT Node Mobility	✓	×	×	✓	×	×	✓
Fog Node Mobility	×	✓	×	×	✓	×	✓

* The references that are marked by × have a two-layer architecture and those with ✓ have a three-layer architecture.

3. Mobility Model and Proposed Offloading Policy

Within this section, we begin by exploring the mobility model of fog nodes, delving into the handover and migration processes involved as fog nodes move. Subsequently, we introduce the M2One policy and our computation-offloading strategy and elaborate on the developed offloading algorithm.

3.1. Mobility Model

This paper delves into the realm of fog node and IoT device mobility. The mobility of IoT devices primarily impacts request generation. As long as a mobile IoT device resides within the designated area, the tasks it generates are routed to the existing fog nodes within that area. When an IoT device departs from the area, it has no bearing on the functionality of the fog environment.

Fog node mobility holds significant importance as these nodes are responsible for processing generated tasks. Any fog node leaving the area without appropriate request management can threaten the system's reliability. Consequently, it is imperative to develop a model for fog node mobility within this context; therefore, we propose a simple model for the mobility of fog nodes. This model divides each layer into domains where a single IoT application is implemented. The mobility of fog nodes is just considered for the first-tier fog nodes. It is not far from reality because the fog nodes located in the upper tiers (e.g., cloudlets, base stations) are stronger in computing power and storage capability and usually have a fixed location.

According to the stated assumptions, we can consider two types of movement for mobile fog nodes: in-domain movement and off-domain movement. Limited energy resources on IoT nodes mandate low-power and short-range wireless technologies (e.g., BLE, ZigBee); hence, their domain is limited to the distance they can cover, and in-domain movement does not lead to disconnection. Off-domain movement causes problems for the IoT node, which has sent its requests directly to the off-domain fog node, and the requests are offloaded to the off-domain fog node by neighbouring fog nodes. To mitigate disconnection problems, we considered an area called the *boundary area* around the domain. This area comprises about 30% of its corresponding domain. The boundary area and mobility procedure are depicted in Figure 2. A mobile fog node will likely leave the domain when it enters the boundary area. Hence, the handover process and migration process begin to avoid missing the requests that already exist in the off-domain fog node.

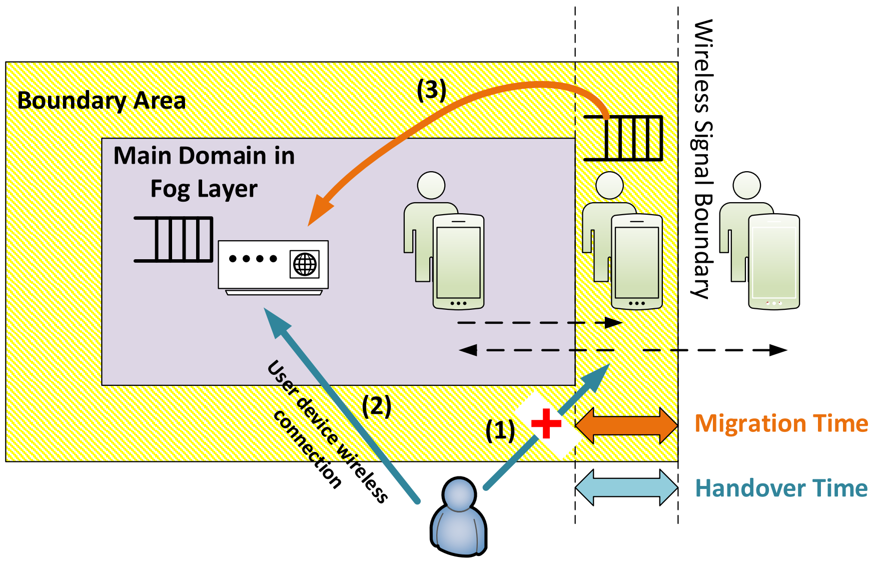


Figure 2. Mobility procedure and boundary area of a specific domain. (1) The IoT node stops sending requests to the off-domain fog node; (2) the IoT node starts a new interaction with a new in-domain fog node; (3) the off-domain fog node sends a copy of the existing request in its processing queue to a new in-domain fog node.

The handover process involves finding a new fog node for the IoT node directly connected to that off-domain fog node. As depicted in Figure 2, in Step (1), the IoT node stops sending requests to that off-domain fog node, and then, in Step (2), it starts a new interaction with a fog node in the domain.

The migration process involves moving a copy of existing requests in the queue of the off-domain fog node to a new fog node in the domain (Step (3) in Figure 2). At this time, neighbouring nodes stop offloading new requests to that off-domain fog node, but existing requests are processed at that off-domain fog node until they leave the boundary area. When that off-domain fog node leaves the boundary area, the fog node that has received the copy of requests starts processing them.

3.2. M2One Policy: A Fog Node Collaboration Policy

In this section, we introduce the M2One policy in which fog nodes collaborate to handle the requests sent from IoT nodes. The M2One policy manages in a many-to-one scenario where all IoT nodes of a specific application offload their requests to one fog node called the coordinator fog node. In other words, in each domain of fog nodes, there is a central node responsible for managing and deciding where to process requests. The first node of the things-to-cloud continuum that receives requests sent from IoT nodes is selected as the coordinator fog node in our scheme. This decision is made based on the fog nodes' computing power, queuing delay, transmission and propagation delay, channel status, and the battery life of the mobile fog nodes. The list of notations that have been used in this section is detailed in the Abbreviation section.

Offloading algorithm: In the proposed policy, we assumed there are j fog nodes F_1, $F_2, \ldots, F_j$, and one of them is the coordinator fog node (F_{j^*}). As the fog network manager, the coordinator fog node knows the network topology. It records fog nodes' location (loc_j), fog nodes' computing power (η_j), and the mobility status of fog nodes in a table, which is called the Coordinator Fog Node (CoFN) table. The coordinator fog node also adds a cloud to its table in case fog nodes cannot fulfil the request objectives (for instance, when fog nodes lack sufficient computing capacity or necessary memory to process a request prior to the deadline). The information in this table is updated when a fog node leaves/enters the domain or a mobile fog node changes its location.

We assumed there are m IoT nodes in the network. These IoT nodes generate i requests $R_1, R_2, \ldots, R_i$. Each request sent from an IoT node m (R_i^m) contains the ID, the location of the IoT node that generated the request (loc_m^i), the type of request that can be heavy ($R_{i:H}^m$) or light ($R_{i:L}^m$), the length of request in bit (len_i), the needed computational resource (ω_i), and the deadline (θ_i). The requests that can be divided into several sub-tasks are considered heavy; otherwise, they are light. The CoFN table information and R_i data are utilised when deciding which fog node to offload a request for processing. Algorithm 1 describes the process of selecting an appropriate node for offloading.

In the first step, the needed computational resources of request (ω_i) are compared with the computing power of all fog nodes (η_j) of the CoFN table. The fog nodes that can fulfil ω_i are added to the *selected nodes* list (Lines 4–6). Then, for all selected nodes, the estimated serving time ($T_{serve(i,j)}^{est.}$) is calculated (Lines 7–8) based on Equation (1).

Algorithm 1 M2One: Proposed Offloading Policy

```
Require: $R_i^m$ {ID, $loc_m^i$, Type, $len_i$, $\omega_i$, $\theta_i$}
Ensure: Selected node for offloading ($F_j$)

 1: function OFFLOADINGPOLICY($R_i^m$)
 2:     Add all fog nodes to CoFN Table
 3:     Add Cloud to CoFN Table
 4:     for all $F_j$ in CoFN Table do
 5:         if $\eta_j \geq \omega_i$ then
 6:             Add $F_j$ to Selected-Nodes
 7:     for all $F_j$ in Selected-Nodes do
 8:         Calculate $T_{serve(i,j)}^{est.}$
 9:         if $T_{serve(i,j)}^{est.} \geq \theta_i$ then
10:             Remove $F_j$ from Selected-Nodes
11:     Sort Selected-Nodes list
12:     Select two nodes with minimum $T_{serve(i,j)}^{est.}$
13:     Remove two selected nodes from Selected-Nodes list
14:     for all node in Two-Selected-Nodes do
15:         if Is it mobile then
16:             Calculate $E_{(i,j)}^{est.}$
17:             Calculate RSSI
18:             if ($E_{(i,j)}^{est.} \leq E_{th}$ OR $RSSI \leq RSSI_{th}$) then
19:                 Remove this mobile node
20:         else
21:             Calculate RSSI
22:             if RSSI $\leq RSSI_{th}$ then
23:                 Remove this node
24:     if Two-Selected-Nodes list is empty then
25:         go to (12)
26:     else
27:         Return node with the least $T_{serve(i,j)}^{est.}$
```

Estimated serving time: The estimated serving time is the required time for serving R_i in F_j. $T_{serve(i,j)}^{est.}$ is a function of the required time to send R_i from the coordinator fog node (F_{j^*}) to F_j ($T_{send(i,j)}$) and the required time to process R_i in F_j ($T_{process(i,j)}$). For simplicity, the required time to send back a response to the IoT node has been omitted in the equations.

$$T^{est.}_{serve(i,j)} = T_{send(i,j)} + T_{process(i,j)} \tag{1}$$

$T_{send(i,j)}$ is the sum of the transmission delay ($T_{trans(j^*,j)}$) and propagation delay ($T_{prop(j^*,j)}$). It is obtained by Equation (2). R_{j^*j} is the channel bit rate between F_{j^*} and F_j; d_{j^*j} is the distance between F_{j^*} and F_j; v is the speed of light.

$$T_{send(i,j)} = T_{trans(i,j)} + T_{prop(i,j)} = \sum_{l}^{j} \frac{len_i}{R_{j^*l}} + \frac{d_{j^*j}}{v} \tag{2}$$

$T_{process(i,j)}$ is the sum of the required time to wait in the F_j queue ($T^j_{queuing}$) and the required time to execute R_i in F_j ($T_{exe(i,j)}$) and is calculated by Equation (3).

$$T_{process(i,j)} = T^j_{queuing} + T_{exe(i,j)} \tag{3}$$

$T^j_{queuing}$ and $T_{exe(i,j)}$ are calculated by Equations (4) and (5), respectively. $\chi^j_{R_{i:L}}$ is the number of light requests in the F_j queue; $\chi^j_{R_{i:H}}$ is the number of heavy requests in the F_j queue; $T_{exe(i^*,j)}$ is the execution time of the running request (R_{i^*}). The running request runs in F_j when an offloaded request R_i arrives.

$$T^j_{queuing} = \chi^j_{R_{i:L}}(T_{exe(i:L,j)}) + \chi^j_{R_{i:H}}(T_{exe(i:H,j)}) + T_{exe(i^*,j)} \tag{4}$$

$$T_{exe(i,j)} = \frac{\omega_i}{\eta_j} \tag{5}$$

After calculating $T^{est.}_{serve(i,j)}$ for all fog nodes, two fog nodes that have met the deadline and have the shortest serving time are selected (Lines 9–13). If the selected node is a mobile fog node, its battery life and channel status are checked (Lines 14–19). Otherwise, the channel status is just checked (Lines 20–23). The battery life is checked based on the remaining energy consumption of each mobile fog node, and the channel status is checked based on the Received Signal Strength Indicator (RSSI). The fog node will remain in the two_selected_nodes list if its $E^{est.}_{(i,j)}$ and RSSI are greater than the threshold that has been considered for each case. Finally, the node with less $T^{est.}_{serve(i,j)}$ is selected as the destination of offloading.

Energy consumption of mobile fog node: The battery life of the mobile fog nodes is a limiting factor in the offloading strategies. Therefore, it is necessary to check the battery status of the mobile fog nodes before offloading requests to them. In this way, we can be sure that the mobile fog node will remain active until the end of the computation. For this purpose, in the M2One policy, a threshold is considered for the remaining energy of the mobile fog nodes (E_{th}). E_{th} is compared with estimated energy ($E^{est.}_{(i,j)}$) (Line 19), and the node that can fulfil the energy requirement will remain in the *two_selected_nodes* list. $E^{est.}_{(i,j)}$ is the remaining energy in F_j after processing R_i. It is a function of the needed energy for queuing requests' execution and transmission, $E^j_{queuing}$, as well as the execution and transmission energy of the running request R_{i^*}, denoted as $E_{exe(i^*,j)}$ and $E_{send(i^*,j)}$, respectively, as described in Equations (6) and (7). E_{init} is the initial energy of the mobile fog nodes before offloading R_i to them.

$$E^{est.}_{(i,j)} = E_{init} - (E^j_{queuing} + E_{exe(i^*,j)} + E_{send(i^*,j^*)}) \tag{6}$$

$$E^{j}_{queuing} = \chi^{j}_{R_{i:L}}(E_{exe(i:L,j)} + E_{send(i:L,j^*)}) + \chi^{j}_{R_{i:H}}(E_{exe(i:H,j)} + E_{send(i:H,j^*)}) + E_{exe(i,j)} + E_{send(i,j^*)} \quad (7)$$

$E_{exe(i,j)}$ is the required energy to execute R_i in F_j. $E_{send(i,j^*)}$ is the required energy to transmit the response from F_j to F_{j^*}. $P_{exe(i,j)}$ is the power consumption of F_j when R_i is executing. It is different for light and heavy requests. P_t is the transmission power.

$$E_{send(i,j^*)} = P_t.T_{send(i,j^*)} \quad (8)$$

RSSI: The received signal strength indicator indicates the signal power of a message received by a node. The proposed offloading policy channel status is checked for the first-tier fog nodes because these nodes usually use low-power wireless communication protocols, which are more vulnerable to noise. The channel status of the selected node is calculated by Equation (9) and, then, compared with a threshold value ($RSSI_{th}$).

In Equation (9), n is the path loss exponent, which varies depending on the environment, d is the distance between the transmitting and receiving devices, and A is the signal strength received from a node located at a reference distance. In Equation (10), P_t is the transmission power and $P_t(d_0)$ is the path loss at a reference distance (d_0).

$$RSSI = A - 10n \log_{10} d \quad (9)$$

$$A = P_t - P_t(d_0) \quad (10)$$

$$E_{exe(i,j)} = P_{exe(i,j)}.T_{exe(i,j)} \quad (11)$$

4. Simulation Setup

In this section, the M2One policy was implemented on a healthcare-monitoring architecture. The simulated application scenario is a healthcare application that collects vital signs, processes them, and then, reports the patient's status. The simulations were carried out in iFogSim [10], which is an often-used simulation tool because of its flexibility.

4.1. Healthcare Monitoring Architecture

The healthcare monitoring system is among the most-popular applications of fog computing. It uses a large number of sensors and requires real-time and low-latency processing. In this paper, we considered a hospital environment, where a patient's clinical data, such as blood pressure, body temperature, blood oxygen level, respiration rate, and ECG signals, are collected by sensors. The collected data are then sent to the cloud for processing, and finally, the patient status (high-risk, normal, low-risk) is sent back to an actuator. The sensors, such as blood pressure and body temperature, produce light requests, but requests that are produced by the ECG tool are heavy. The heavy request usually needs pre-processing and can be divided into several light requests.

To evaluate the effectiveness of fog computing in this scenario, we used the architecture shown in Figure 3. In the first layer of this architecture, there were four static patients and three mobile patients. The arrival times of these mobile patients were chosen randomly, and their duration within the simulated area was determined randomly, spanning from 2% to 10% of the simulation's total time. They also walked in a random direction with an average speed of 1 m per second. Four light sensors, one heavy sensor, and one actuator were considered for each patient. The patients were in a room with an area of 10.5 m $\times$ 6.5 m, which is the size of a typical four-bed hospital room. There were two tiers in the fog layer. In the first tier, there were four fog nodes, where two of them were fixed, and the two others were mobile. There was a static fog node in the second tier. Table 2 shows the overall configurations of the devices and connection links used in our simulations.

4.2. Mobility Scenario

As mentioned in the previous subsection, two mobile fog nodes were in the first tier of the fog layer. During the simulation, the first mobile fog node left the application domain, and after a while, a new mobile fog node entered and remained in the domain until the end of the simulation. The average speed of a mobile fog node was considered 1 m/s, and they moved in a predefined direction. The reason for considering a predefined direction was to ensure that the mobile fog node moved in the different parts of the domain. As shown in Figure 3, there was a boundary area around the first tier of the fog layer. This area included about 30% of the domain.

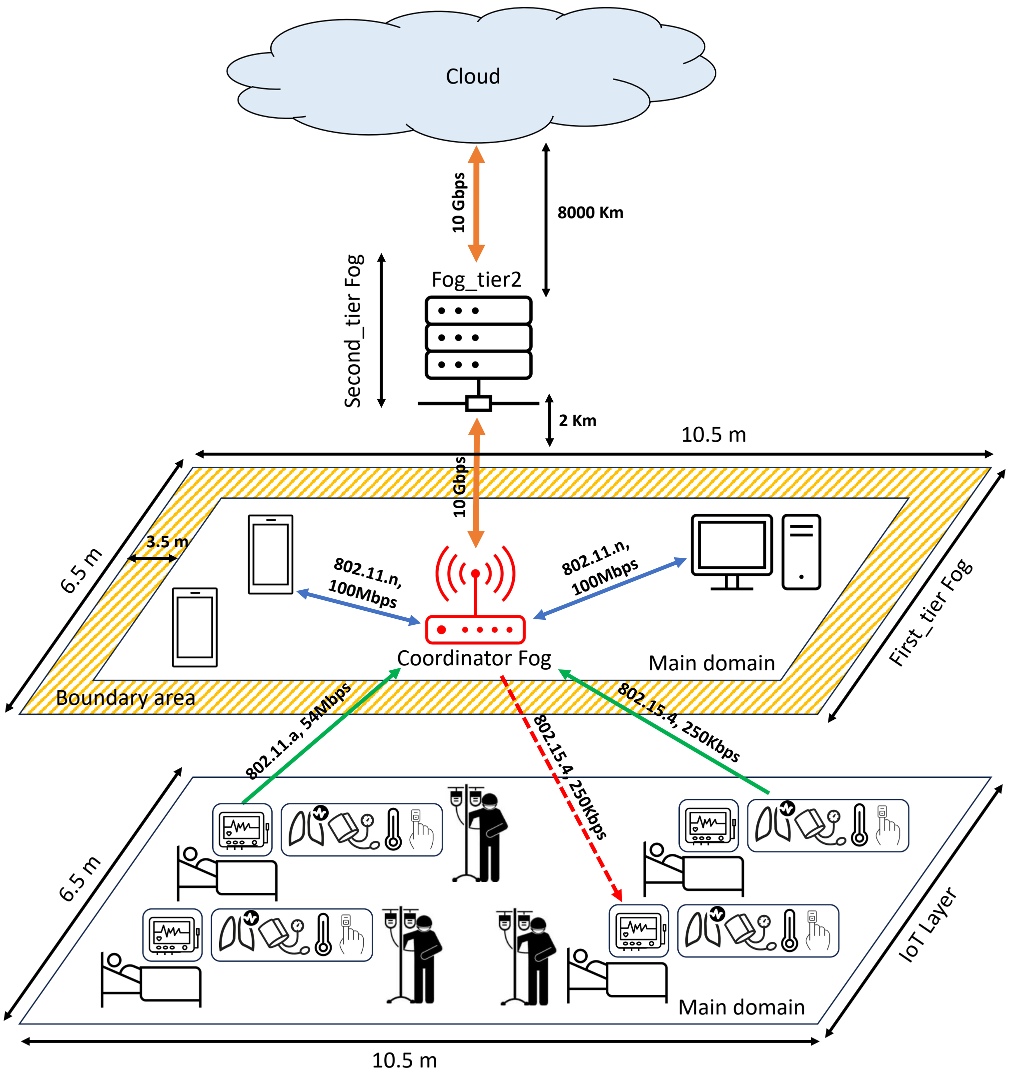


Figure 3. Simulated architecture for healthcare application.

When a mobile fog node entered the boundary area, the coordinator fog node stopped offloading a new request to it. That mobile fog node also sent a copy of its queuing requests to the coordinator fog node. When the mobile fog node left the boundary area and quit the domain, the coordinator fog node offloaded that mobile fog node's corresponding requests to other fog nodes with a priority. In other words, the coordinator fog node calculated the time each request spent in that mobile fog node queue, subtracted this time from the deadline, and considered this new value as a new deadline. When a new mobile fog node entered the boundary area from outside the domain, the coordinator fog node added it to its table, but offloaded no request to this new mobile fog node until it entered the main domain of the application.

Table 2. Simulation parameters.

Channel Bit Rate			
$Sensor_{Light}$-F_{j^*}	250 Kbps	$Sensor_{Heavy}$-F_{j^*}	54 Mbps
F_{j^*}-MobileFog	100 Mbps	F_{j^*}-StaticFog	100 Mbps
F_{j^*}-Fog_{tier2}	10 Gbps	Fog_{tier2}-Cloud	10 Gbps
Requests			
	ω_i	len_i	$P_{exe(i,j)}$
Light Request	100	100 B	100 mW
Heavy Request	800	80 KB	400 mW
Constant Values			
P_t	100 mW	v	3×10^8
E_{init}	18 Wh	d_0	1 m
E_{th}	5 Wh	θ_i	1 s
$RSSI_{th}$	−80 dB	n	U [1.6–1.8]
# of Light Sensors/Patient	4	# of Heavy Sensors/Patient	1

4.3. Operation Modes and Configurations

To evaluate the advantages of the proposed offloading policy, we conducted a comparative analysis of light/heavy requests' service delay and network usage across three distinct operational modes. The first mode, referred to as Cloud, simulated traditional cloud computing, where all requests were routed to cloud servers. The second mode, denoted as M2One_FogCloud, implemented the M2One policy within a three-layer fog computing framework. Here, offloaded tasks can be executed in the fog or cloud layers. The third mode, named M2One_FogOnly, involved the distribution of requests between the first and second tiers of the fog layer based on the M2One policy. This mode employed a two-layer fog computing framework, with all IoT-generated requests exclusively processed within the fog layer.

In the simulation setup of this study, we conducted an extensive investigation into the impact of varying fog node computing power and the number of patients on the delay and network usage metrics. To facilitate this analysis, we designed two distinct sets of configurations. In the first set of configurations, we modified the computing capacity of the fog nodes, as detailed in Table 3. This allowed us to explore how changes in the fog node computational capabilities influenced our experimental outcomes.

Table 3. The configurations of the different fog node computing power capacity scenarios.

	Config. 1	**Config. 2**	**Config. 3**
Cloud_MIPS	44,800	44,800	44,800
Second_tier Fog_MIPS	2240	4480	8960
First_tier Fogs_MIPS	1400	2800	5600

In the second set of configurations, we maintained a fixed computing capacity for the fog nodes while varying the number of patients, as specified in Table 4. This adjustment altered the computational demands and prompted us to adapt the simulated area size. To align with the standard dimensions of hospital rooms, we adjusted the simulated area size based on the number of patients. Specifically, we adhered to the standard size for a bed in a multiple-bed room, which is 7.48 m^2, and ensured a requisite 2.5 m spacing between beds [27].

Table 4. The configurations of the different number of patients scenarios.

		Varying Static Patients		
Number of Static Patients	2	4	6	8
		Varying Mobile Patients		
Number of Mobile Patients	0	4	8	12

Additionally, we conducted an in-depth examination to assess the impact of the simulation time on our results. Throughout these investigations, we varied the simulation time from 600 to 3600 s. Notably, our findings revealed that altering the simulation time did not yield significant variations in the results when calculating the average delay. Furthermore, for the request's emit intervals, we opted for a 2 s interval for heavy requests and a 10 s interval for light requests. This deliberate choice of intervals afforded ample time to execute critical actions, particularly in emergencies.

5. Experimental Results

This section presents the simulation results of the different configurations. These results are presented for different fog node computing power scenarios and different numbers of patient scenarios.

5.1. The Effect of Fog Node Computing Power on Evaluation Metrics

In this paper, we conducted a series of simulations to investigate the impact of fog node computing power on average delay and network usage within various operation modes. Our default configuration, referred to as *Config. 2* (Second_tier Fog_MIPS: 4480, First_tier Fog_MIPS: 2800), served as the baseline for our analysis. We explored the consequences of both increasing and decreasing the computing power capacity in each operation mode and the disparities in the average delay and network usage between different operation modes.

Generally, as shown in Figure 4, we observed a significant improvement in our evaluation metrics, such as the average delay and network usage, when the fog node computing power was increased. This improvement was particularly pronounced in the M2one_FogCloud and M2one_FogOnly operation modes. In the M2one_FogCloud mode, by doubling the computing power of fog nodes, we observed a notable decrease in average delay for both light and heavy requests, amounting to approximately 45% and 49%, respectively.

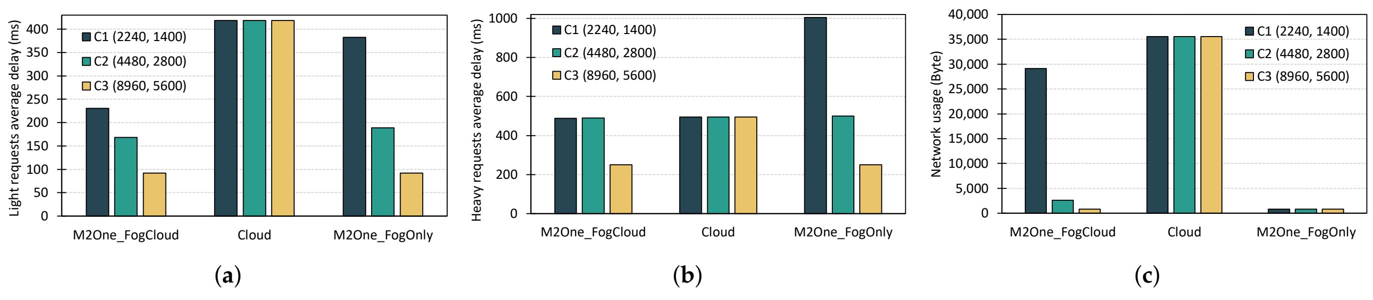


Figure 4. Average delay and network usage in the different fog node computing power scenarios. Config. # = C# (Fog_{tier2}_MIPS, Fog_{tier1}_MIPS). (**a**) Light requests' average delay; (**b**) heavy requests' average delay; (**c**) network usage.

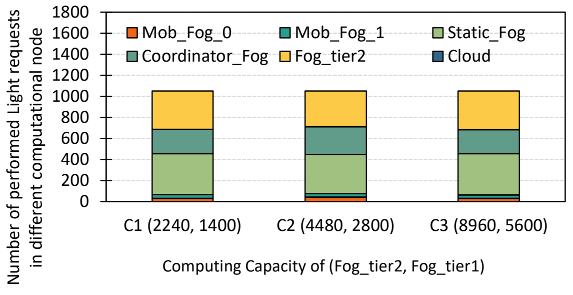
Furthermore, a decrease in fog node computing power by half led to a significant increase in the average delay of both light and heavy requests, with increases of approximately 38% and 1%, respectively. Notably, the heavy request average delay was comparable between C1 (Second_tier Fog_MIPS: 2240, First_tier Fog_MIPS: 1400) and C2 (Second_tier Fog_MIPS: 4480, First_tier Fog_MIPS: 2800); however, an interesting divergence emerged.

As shown in Figure 5b, in C1, about 82% of requests were routed to the cloud, while in C2, only 7% of requests were processed in the cloud, with the remainder being handled in the fog layer. This observation suggested that the C2 values represented a threshold for computing power in our simulated scenario. Although all requests in C2 were processed in the fog layer, the fog nodes were not sufficiently powerful, resulting in an average delay nearly identical to that of the cloud. This indicated that the computation delay of fog nodes in C2 was comparable to the transmission delay to the cloud.

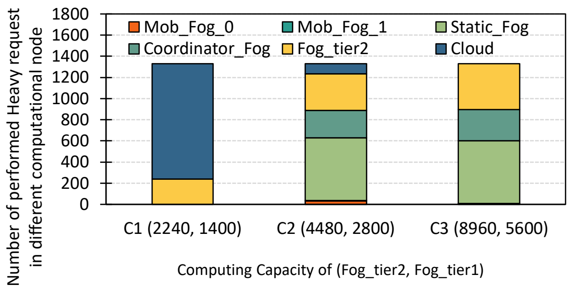
In the M2one_FogOnly mode, we found that the network performance was highly dependent on the computing power of the fog nodes. Figure 4 illustrates that the average delay of both light and heavy requests exhibited significant variations with changes in fog node capacity.

Concerning network usage, we observed distinct patterns across the various operation modes (Figure 4c). Notably, the highest network usage was observed in the Cloud mode, where all requests were forwarded to the cloud data centres. Conversely, the M2one_FogOnly mode exhibited the lowest network usage since no requests were sent to the cloud, and all processing occurred locally, resulting in minimal bandwidth usage.

In the M2one_FogCloud mode, we observed a noteworthy increase in network usage when the fog node capacity was reduced by half. This increase, approximately 12-times, can be attributed to routing most heavy requests to the cloud, as previously discussed. Although C1 and C2 exhibited similar average delay values for heavy requests, their network usage patterns diverged significantly. C2 experienced substantially lower network usage due to the efficient processing of requests within the fog layer. Furthermore, doubling the computing power of fog nodes in the M2one_FogCloud mode resulted in a remarkable 30% reduction in network usage. This reduction aligned network usage with the M2one_FogOnly mode, as no requests were sent to the cloud in this configuration.



(**a**)



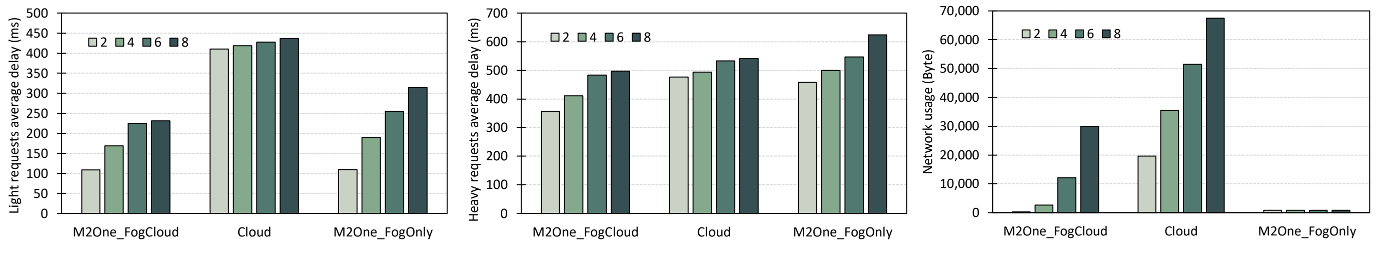
(**b**)

Figure 5. Number of requests performed in each computational node in the M2One_FogCloud mode in the different fog nodes capacity scenarios. (**a**) Light requests; (**b**) heavy requests.

5.2. The Effect of Number of Static/Mobile Patients on Evaluation Metrics

To investigate the impact of increasing the number of requests offloaded to fog nodes, we focused on patient load scenarios ranging from 2 to 8 individuals. To ensure a realistic representation, we adjusted the size of the hospital rooms to align with standard dimensions.

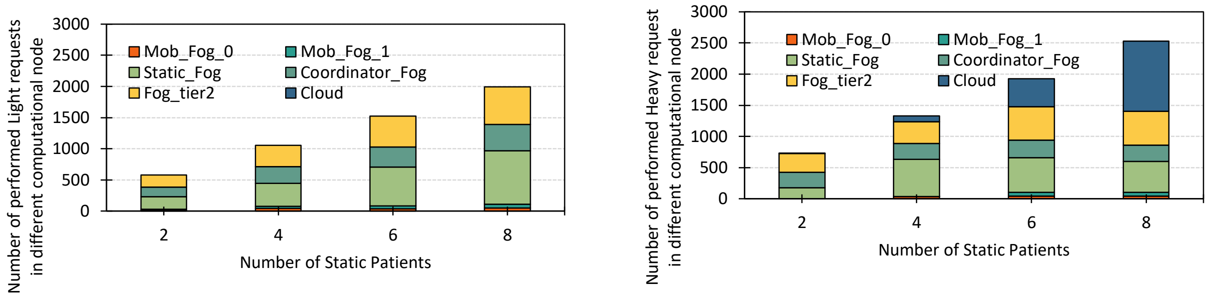
As depicted in Figure 6, our findings indicated a notable trend when increasing the number of patients in the M2one_FogCloud mode. Specifically, we observed that, as the number of patients increased from 2 to 8, the average delay for both light and heavy requests increased by approximately 100% and 40%, respectively. A similar pattern emerged in the M2one_FogOnly mode, with an increase of approximately 185% for light requests and 36% for heavy requests. This discrepancy in average delay between M2one_FogOnly and M2one_FogCloud can be attributed to the coexistence of heavy requests in the fog layer in M2one_FogOnly, leading to extended processing times for light requests. However, an intriguing observation arose when we explored the performance of light and heavy requests for patient counts exceeding four.



(**a**) (**b**) (**c**)

Figure 6. Average delay and network usage in the different number of static patients scenarios. (**a**) Light requests' average delay; (**b**) heavy requests' average delay; (**c**) network usage.

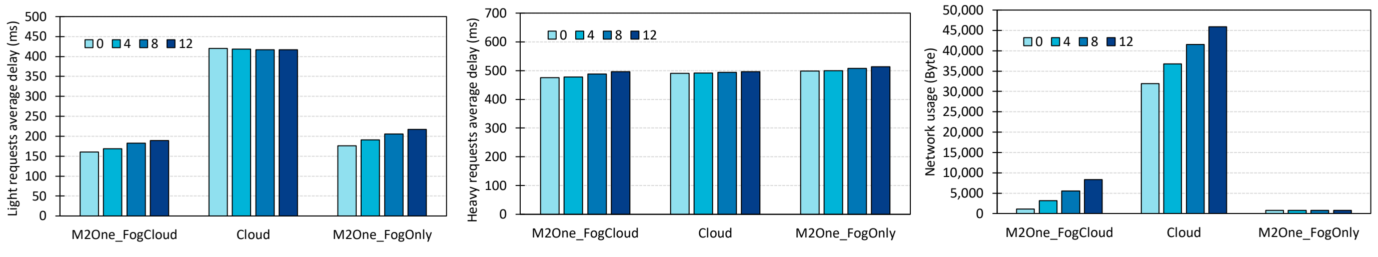
In these cases, as illustrated in Figure 7b, all additional requests beyond four patients were forwarded to the cloud due to the limitations of the existing fog nodes. Consequently, heavy requests' average delays for 6 and 8 patients became almost identical. As discussed earlier, the available fog node capacity can adequately support up to four patients; surpassing this threshold caused heavy request delays in the M2One_FogCloud mode to mimic those in the Cloud mode.



(**a**) (**b**)

Figure 7. Number of requests performed in each computational node in the M2One_FogCloud mode in the different number of static patient scenarios. (**a**) Light requests; (**b**) heavy requests.

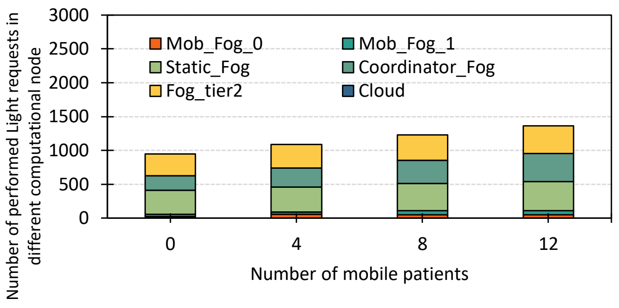
An essential consideration that emerged from our study was the necessity to tailor fog node computing capacity to the requested volume of a given area. In certain environments, such as hospitals, we had static patients and mobile patients moving between various locations. To examine the impact of mobile patients, we introduced a scenario with four static patients in a room and varied the number of mobile patients from 0 to 12. Our results, presented in Figure 8, indicated that the presence of mobile patients had a limited effect on the average delay of both light and heavy requests. Mobile patients generated requests for only a fraction of their time within an area, and their sporadic presence ensured that the delay remained relatively constant across different mobile patient counts.



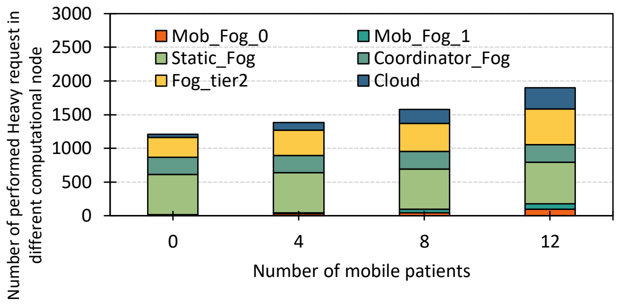
(**a**) (**b**) (**c**)

Figure 8. Average delay and network usage in the different number of mobile patients scenarios. (**a**) Light requests' average delay; (**b**) heavy requests' average delay; (**c**) network usage.

Interestingly, when we increased the mobile patient count to 12, the total number of light and heavy requests generated equalled the number generated when there were 6 static patients. However, Figures 7b and 9b highlight a crucial distinction: the percentage of heavy requests sent to the cloud in the presence of 12 mobile patients was approximately 10% less than the percentage when 6 static patients were present. This suggested that distributing request generation over time, instead of generating requests within a specific time window, can significantly alleviate the load on fog nodes and expedite request processing.



(**a**)



(**b**)

Figure 9. Number of requests performed in each computational node in the M2One_FogCloud mode in the different number of mobile patient scenarios. (**a**) Light requests; (**b**) heavy requests.

Turning our attention to network usage, as shown in Figure 6c, an increase in static patient count led to a corresponding rise in network utilisation. Specifically, when we increased the number of static patients from 4 to 8, network usage surged by approximately 11-times in the M2One_FogCloud mode and 3.5-times in the Cloud mode. Notably, Figure 8c reveals a different trend when we introduced 12 mobile patients, resulting in network usage that was approximately 30% lower than that observed with 6 static patients.

5.3. Results Discussion

In summary, our study underscored the significant influence of network parameters on the performance of a fog computing network. However, within the context of our simulations, the proposed method, when applied within a three-layer architecture comprising the IoT layer, fog layer, and cloud layer, consistently outperformed traditional cloud computing, due to over 90% of heavy requests and all light requests being processed within the fog layer. We observed an impressive improvement of approximately 30% in average delay and a significant reduction of 90% in network usage compared to cloud computing. Notably, further enhancements could be achieved by deploying more-robust fog nodes within the fog layer of our network infrastructure. If the fog layer is sufficiently empowered to manage all types of requests, an additional 60% decrease in average delay, alongside a substantial reduction in network usage, could be observed.

Applications vary in their time sensitivity and computational intensity, with some requiring both aspects. Consequently, the selection and quantity of fog nodes should be tailored to meet specific application requirements.

6. Conclusions

The dynamic landscape of IoT applications coupled with the rise of fog computing offers a substantial opportunity to enhance Quality of Service (QoS) significantly. Computation offloading is a vital bridge between the IoT and fog nodes, strategically distributing computational tasks across multiple resource-constrained processing nodes. Within this realm, our contribution, the Many-to-One (M2One) policy, stands as a strategic offloading approach that considers pivotal factors, including queuing and transmission delays, communication channel status, and the limited battery life of mobile fog nodes. Implementing the M2One policy within a healthcare-monitoring architecture showcased remarkable adaptability to the ever-changing mobility patterns of patients and fog nodes.

Through extensive experimentation, we carefully studied changes in the network parameters, specifically examining changes in the fog nodes' computing capacities and the volume of generated requests. These investigations shed light on the consequential impact on both the average delay and network usage. Our performance analysis illustrated that our proposed policy led to an approximate 30% enhancement in average delay in comparison to cloud computing. Additionally, a considerable 90% reduction in network usage was observed, underscoring the effectiveness of our framework in meeting the demanding latency requirements of healthcare applications. While the primary focus of this study revolved around healthcare applications, the implications of our results extend across various sectors that generate copious real-time data, such as video surveillance, autonomous vehicles, and sports analytics applications.

Moving forward, future work will concentrate on enhancing the precision of mobile fog nodes and user location determination. We aim to propose a mobility prediction approach to forecast the next locations of mobile nodes, further optimising task allocation and resource utilisation. Additionally, our focus will extend to exploring adaptive mechanisms to accommodate diverse IoT applications, ensuring robustness and scalability in fog computing environments.

Author Contributions: Conceptualisation, F.S., S.H. and Y.C.L.; formal analysis, F.S., S.H. and Y.C.L.; investigation, F.S., A.R., A.P., S.H. and Y.C.L.; methodology, F.S., S.H. and Y.C.L.; software, F.S. and A.R.; validation, F.S., A.R., A.P., S.H. and Y.C.L.; visualisation, F.S. and A.P.; writing—original draft, F.S. and A.P.; writing—review and editing, F.S., A.P., S.H. and Y.C.L.; project administration, Y.C.L.; supervision, S.H. and Y.C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data is unavailable due to privacy or ethical restrictions.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

F_j	Fog node j
F_{j^*}	Coordinator fog node
R_i^m	Request i generated by the IoT node m
$R_{i:L}^m$	A light request
$R_{i:H}^m$	A heavy request
R_{i^*}	The request that is running on F_j
len_i	Request size of R_i^m (in bits)
ω_i	Needed computational resources of R_i^m
θ_i	Deadline (max latency required) of R_i^m
η_j	Computing power of F_j (in MIPS)
$T_{serve(i,j)}^{est.}$	The estimated time for serving R_i in F_j
$T_{send(i,j)}$	The required time to send R_i to F_j
$T_{trans(j^*,j)}$	Transmission delay to send R_i from F_{j^*} to F_j
$T_{prop(j^*,j)}$	Propagation delay to send R_i from F_{j^*} to F_j
$T_{process(i,j)}$	The required time to process R_i in F_j
$T_{queuing}^j$	The required time to wait in the F_j queue
$T_{exe(i,j)}$	The required time to execute R_i in F_j
$\chi_{R_{i:L}}^j$	Number of light request in the F_j queue
$\chi_{R_{i:H}}^j$	Number of heavy request in the F_j queue
R_{j^*j}	Channel bit rate between F_{j^*} and F_j

$E^{est.}_{(i,j)}$	The remaining energy in F_j after processing R_i
$E^{j}_{queuing}$	The queuing requests' processing energy
$E_{exe(i,j)}$	The required energy to execute R_i in F_j
$E_{send(j,j^*)}$	The transmission energy
$P_{exe(i,j)}$	The power consumption of F_j during execution
P_t	The transmission power

References

1. IDC. The Digitization of the World From Edge to Core. Available online: https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf (accessed on 20 November 2023)
2. Ravandi, B.; Papapanagiotou, T. A self-learning scheduling in cloud software defined block storage. In Proceedings of the IEEE 10th International Conference on Cloud Computing, Honolulu, HI, USA, 25–30 June 2017; pp. 415–422.
3. Yousefpour, A.; Fung, C.; Nguyen, T.; Kadiyala, K.; Jalali, F.; Niakanlahiji, A.; Kong, J.; Jue, J. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J. Syst. Archit.* **2019**, *98*, 289–330. [CrossRef]
4. Kong, L.; Tan, J.; Huang, J.; Wang, S.; Jin, X.; Zeng, P.; Khan, M.; Das, S. Edge-computing-driven internet of things: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–41. [CrossRef]
5. CISCO. Cisco Fog Computing Solutions: Unleash the Power of the Internet of Things. 2015. Available online: https://docplayer.net/20003565-Cisco-fog-computing-solutions-unleash-the-power-of-the-internet-of-things.html (accessed on 20 Nov 2023).
6. OpenFog. Openfog Reference Architecture for Fog Computing. 2017. Available online: https://www.iiconsortium.org/pdf/OpenFog_Reference_Architecture_2_09_17.pdf (accessed on 20 Nov 2023).
7. Lin, H.; Zeadally, S.; Chen, Z.; Labiod, H.; Wang, L. A survey on computation offloading modeling for edge computing. *J. Netw. Comput. Appl.* **2020**, *169*, 102781. [CrossRef]
8. Sharifi, F.; Hessabi, S.; Rasaii, A. The Effect of Fog Offloading on the Energy Consumption of Computational Nodes. In Proceedings of the 4th International Symposium on Real-Time and Embedded Systems and Technologies, Tehran, Iran, 30–31 May 2022; pp. 1–6.
9. Sharifi, F.; Rasaii, A.; Honarmand, M.; Hessabi, S.; Choon Lee, Y. Mobility-Aware Fog Offloading. In Proceedings of the 24th Asia-Pacific Network Operations and Management Symposium (APNOMS), Sejong, Republic of Korea, 6–8 September 2023; pp. 24–29.
10. Gupta, H.; Vahid-Dastjerdi, A.; Ghosh, S.; Buyya, R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Softw. Pract. Exp.* **2017**, *47*, 1275–1296. [CrossRef]
11. Zhu, C.; Pastor, G.; Xiao, Y.; Li, Y.; Ylae-Jaeaeski, A. Fog following me: Latency and quality balanced task allocation in vehicular fog computing. In Proceedings of the 15th Annual IEEE International Conference on Sensing, Communication, and Networking, Hong Kong, China, 11–13 June 2018; pp. 1–9.
12. Etemadi, M.; Ghobaei-Arani, M.; Shahidinejad, A. Resource provisioning for IoT services in the fog computing environment: An autonomic approach. *Comput. Commun.* **2020**, *161*, 109–131. [CrossRef]
13. Jin, Y.; Lee, H. On-Demand Computation Offloading Architecture in Fog Networks. *Electronics* **2019**, *8*, 1076. [CrossRef]
14. Hussain, M.; Bed, M. CODE-V: Multi-hop computation offloading in vehicular fog computing. *Future Gener. Comput. Syst.* **2021**, *116*, 86–102. [CrossRef]
15. Farahbakhsh, F.; Shahidinejad, A.; Ghobaei-Arani, M. Context-aware computation offloading for mobile edge computing. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *14*, 5123–5135. [CrossRef]
16. Li, K. Heuristic computation offloading algorithms for mobile users in fog computing. *ACM Trans. Embed. Comput. Syst.* **2021**, *20*, 11. [CrossRef]
17. Cha, N.; Wu, C.; Yoshinaga, T.; Ji, Y.; Yau, K. Virtual edge: Exploring computation offloading in collaborative vehicular edge computing. *IEEE Access* **2021**, *9*, 37739–37751. [CrossRef]
18. Bozorgchenani, A.; Disabato, S.; Tarchi, D.; Roveri, M. An energy harvesting solution for computation offloading in Fog Computing networks. *Comput. Commun.* **2020**, *160*, 577–587. [CrossRef]
19. Zhou, S.; Jadoon, W.; Shuja, J. Machine learning-based offloading strategy for lightweight user mobile edge computing tasks. *Complexity* **2021**, *2021*, 6455617. [CrossRef]
20. Hou, X.; Ren, Z.; Wang, J.; Zheng, S.; Cheng, W.; Zhang, H. Distributed fog computing for latency and reliability guaranteed swarm of drones. *IEEE Access* **2020**, *8*, 7117–7130. [CrossRef]
21. Deng, X.; Sun, Z.; Li, D.; Luo, J.; Wan, S. User-centric computation offloading for edge computing. *IEEE Internet Things J.* **2021**, *8*, 12559–12568. [CrossRef]
22. Li, C.; Cai, Q.; Zhang, C.; Ma, B.; Luo, Y. Computation offloading and service allocation in mobile edge computing. *J. Supercomput.* **2021**, *77*, 13933–13962. [CrossRef]
23. Qiu, Y.; Zhang, H.; Long, K. Computation offloading and wireless resource management for healthcare monitoring in fog-computing-based internet of medical things. *IEEE Internet Things J.* **2021**, *8*, 15875–15883. [CrossRef]
24. Kuang, Z.; Ma, Z.; Li, Z.; Deng, X. Cooperative computation offloading and resource allocation for delay minimization in mobile edge computing. *J. Syst. Archit.* **2021**, *118*, 102167. [CrossRef]
25. Zhu, Q.; Si, B.; Yang, F.; Ma, Y. Task offloading decision in fog computing system. *China Commun.* **2017**, *14*, 59–68. [CrossRef]

26. Yousefpour, A.; Ishigaki, G.; Gour, R.; Jue, J. On reducing IoT service delay via fog offloading. *IEEE Internet Things J.* **2018**, *5*, 998–1010. [CrossRef]
27. Cahnman, S. Design Guidelines for Short-Stay Patient Units. 2017. Available online: https://www.hfmmagazine.com/articles/2841-design-guidelines-for-short-stay-patient-units (accessed on 20 Nov 2023)