An Adaptive SDN-Based Load Balancing Method for Edge/Fog-Based Real-Time Healthcare Systems

Ahmed M. Jasim, *Member, IEEE*, and Hamed Al-Raweshidy, *Senior Member, IEEE*

***Abstract*—Edge/fog computing has gained significant popularity as a computing paradigm that facilitates real-time applications, especially in healthcare systems. However, deploying these systems in real-world healthcare scenarios presents technical challenges, among which load balancing is a key concern. Load balancing aims to distribute workloads evenly across multiple nodes in a network to optimize processing and communication efficiency. This article proposes an adaptive load-balancing method that combines the strengths of static and software-defined networking (SDN)-based load balancing algorithms for edge/fog-based healthcare systems. A new algorithm called load balancing of optimal edge-server placement (LB-OESP) is proposed to balance the workload statically in the systems, followed by the presentation of an SDN-based greedy heuristic (SDN-GH) algorithm to manage the data flow dynamically within the network. The LB-OESP algorithm effectively balances workloads while minimizing the number of edge servers required, thereby improving system performance and saving costs. The SDN-GH algorithm leverages the benefits of SDN to dynamically balance the load and provide a more efficient system. Simulation results demonstrate that the proposed method provides an adaptive load-balancing solution that takes into consideration changing network conditions and ensures improved system performance and reliability. Furthermore, the proposed method offers a 12% reduction in system latency and up to 28% lower deployment costs compared to the previous studies. The proposed method is a promising solution for edge/fog-based healthcare systems, providing an efficient and cost-effective approach to managing workloads.**

***Index Terms*—Edge/fog, healthcare, load balancing, software-defined networking (SDN).**

I. Introduction

THE integration of healthcare with technology has led to the development of healthcare systems that aim to provide real-time monitoring and effective management of patient health. Edge and fog computing have emerged as promising solutions for healthcare applications due to their ability to handle large amounts of data, provide low-latency communication, and support real-time decision making [1].

Manuscript received 18 July 2023; revised 28 December 2023 and 30 March 2024; accepted 13 May 2024. Date of publication 31 May 2024; date of current version 20 June 2024. This work was supported by Brunel University London, U.K. *(Corresponding author: Hamed Al-Raweshidy.)*

Ahmed M. Jasim is with the Department of Electronic & Electrical Engineering, Brunel University London, UB8 3PH London, U.K., and also with the Department of Computer Engineering, College of Engineering, University of Diyala, Baqubah, Iraq (e-mail: ahmed.jasim@brunel.ac.uk, ahmed.1985m@yahoo.com, ahmed.1985m@uodiyala.edu.iq).

Hamed Al-Raweshidy is with the Department of Electronic & Electrical Engineering, Brunel University London, UB8 3PH London, U.K. (e-mail: Hamed.Al-Raweshidy@brunel.ac.uk).

Digital Object Identifier 10.1109/JSYST.2024.3402156

However, the deployment of these systems in real-world healthcare scenarios requires addressing several technical challenges, one of which is load balancing. Load balancing refers to the distribution of workloads evenly among multiple nodes in a network to ensure efficient processing and communication [2]. This approach has several key benefits, including increased system efficiency, faster performance, and lower latency. By reducing the load on each server, load balancing minimizes the risk of network failures and improves the overall responsiveness of applications by distributing the workload evenly [3]. Furthermore, load balancing increases the availability of systems to consumers, making it an essential component for applications in fields such as healthcare and weather forecasting that require a reliable load-balancing algorithm to introduce new features over time [4]. In edge and fog-based healthcare systems, efficient load balancing is critical to avoid overloading of some nodes, which can result in delays and degraded system performance.

There are two main types of load-balancing algorithms: static and dynamic. Static load-balancing algorithms divide tasks without considering the current state of the servers. These algorithms use predefined information such as the execution costs and/or arrival times of tasks and distribute tasks according to a predetermined strategy, such as round-robin or client-side random. While static load balancing can be quickly set up, it can be inefficient and unable to adapt to short-term fluctuations in loads. Dynamic load-balancing algorithms, on the other hand, make decisions during execution based on the current state of the servers, including their health, workload, and availability. They monitor the health of each server through routine health checks and redirect traffic from overloaded or underperforming servers to those that are less used. This keeps the distribution balanced and effective. However, these methods can be more challenging to set up and often have high computational overhead, which makes them impractical for large-scale edge/fog-based systems with limited resources [5], [6], [7], [8].

SDN-based load-balancing algorithms are a type of dynamic load-balancing algorithm that utilize software-defined networking (SDN) capabilities to manage and distribute network traffic. These algorithms provide centralized control and real-time visibility of the network, enabling dynamic reconfiguration in response to changes in network conditions and traffic patterns. This results in a more flexible and efficient system than traditional hardware-based load-balancing methods. By utilizing SDN technology, SDN-based load balancing algorithms can offer a more dynamic and effective way to balance network traffic and optimize network performance [9], [10], [11].

The growing global population has placed significant strain on urban healthcare systems, resulting in 2.6 million annual fatalities attributed to inadequate management [12], [13]. These challenges are further compounded by inefficiencies in patient transfers and fragmented care, leading to suboptimal health outcomes and increased burden on healthcare providers. For instance, delays in accessing specialized care and difficulties in coordinating treatment across multiple providers hinder the delivery of timely and effective healthcare services.

Previous studies [24], [25] have explored the potential of edge/fog computing to address these challenges by establishing healthcare systems for monitoring patients in need. However, one significant obstacle faced by such systems is the increased response latency experienced as the workload grows, primarily due to inadequate load balancing in the underlying algorithms. With urban populations expected to continue growing rapidly, urgent action is imperative to adapt healthcare systems and ensure they can meet the evolving needs of urban residents. Integrating edge/fog computing solutions with efficient load-balancing mechanisms holds great promise in improving healthcare delivery efficiency, enhancing patient outcomes, and alleviating the strain on healthcare providers. This study aims to enhance the system by incorporating load-balancing mechanisms into the algorithms. This strategic improvement seeks to optimize resource utilization, minimize response time, and build upon the achievements of the previous study. This article presents a novel load-balancing method that integrates the strengths of static and SDN-based approaches, offering a practical and scalable solution to the load-balancing challenge in edge/fog-based healthcare systems. The contributions of this article are as follows.

1) A load-balancing framework is introduced for healthcare systems in urban areas, leveraging edge computing, and SDN technology.
2) The load balancing of optimal edge-servers placement (LB-OESP) algorithm is proposed, an enhanced version of the OESP algorithm proposed in [24]. While the OESP algorithm achieved its primary objectives successfully, it might encounter challenges, especially in heavy loads, due to a lack of emphasis on load balancing. Acknowledging this drawback, the LB-OESP algorithm efficiently selects optimal locations for placing edge servers (ESs), surpassing the approach in [24]. In addition, it ensures a balanced connection of other facilities to these servers, aiming for an optimal distribution of the expected load.
3) The SDN-greedy heuristic (SDN-GH) algorithm is proposed as an SDN-based approach. This algorithm dynamically balances the load and facilitates efficient data offloading within the network.
4) Simulation-based evaluations are conducted to assess the performance of the proposed algorithms. The results demonstrate favorable outcomes, including a cost-effective system and reduced latency compared to previous work.

Paper Organization: The structure of this article is organized as follows. In Section II, we review the related work. Sections III and IV present the proposed system model and problem formulation. In Section V, we introduce the LB-OESP and the SDN-GH algorithms. The performance evaluation of the proposed algorithms is discussed in Section VI, where we present and analyze the results. Finally, in Section VII, we draw conclusions based on the findings of this study and discuss potential avenues for future research.

II. Related Work

Edge computing has garnered significant attention for enhancing cloud computing systems by enabling data processing at the network edge, closer to the data source. The advantages offered by edge computing, such as reduced latency and improved user experience, have sparked substantial research interest, resulting in a significant body of literature in this field. In this section, we review recent advancements in edge computing and load balancing, with a particular focus on the crucial issue of load balancing in edge/fog-based applications due to the growing demand for real-time data processing and low-latency communication.

Various load-balancing approaches have been proposed to address this challenge. For instance, Chen et al. [14] proposed a task allocation model to address load balancing at the server level in fog computing. By treating tasks offloaded by other servers as a single large aggregation task, they calculated the completion time of large aggregation tasks on each server. These studies demonstrate the diverse strategies proposed for load balancing in edge/fog-based environments. Wang et al. [15] developed a distributed city-wide traffic management system and proposed an offloading algorithm for real-time traffic management in fog-based Internet of Vehicle (IoV) systems. Their focus on minimizing the average response time of the Traffic Management Server highlights the importance of efficient offloading and real-time management in IoV environments. Ning et al. [16] investigated a joint computation offloading, power allocation, and channel assignment scheme for 5G-enabled traffic management systems. Their integrated approach aimed to enhance system performance and efficiency, showcasing the growing research interest in load-balancing techniques for emerging network paradigms.

A dynamic load-balancing approach with multiple objectives was proposed by Cabrera et al. [17]. Their dynamic load balancing approach, utilizing the Ull multiobjective framework, aims to optimize application performance and resource efficiency. By separating metric gathering, objective functions, and load balancing algorithms, they provide a flexible and cost-effective solution. Similarly, Li et al. [18] addressed the load-balancing problem by proposing a strategy for task allocation based on intermediary nodes. Their multistep process, involving classification, evaluation, and task assignment, contributes to improved load balancing and resource utilization.

Considering the resource optimization aspect, Nayyer et al. [19] developed the load balancing for resource optimization approach. It tackles both resource scarcity and under-provisioning issues in cloudlets, maximizing resource utilization at the cloudlet level. Their approach ensures stable resource utilization without compromising application performance, demonstrating the potential for enhancing resource efficiency in edge computing systems. Similarly, He et al. [20] introduced an enhanced constrained particle swarm optimization algorithm within the

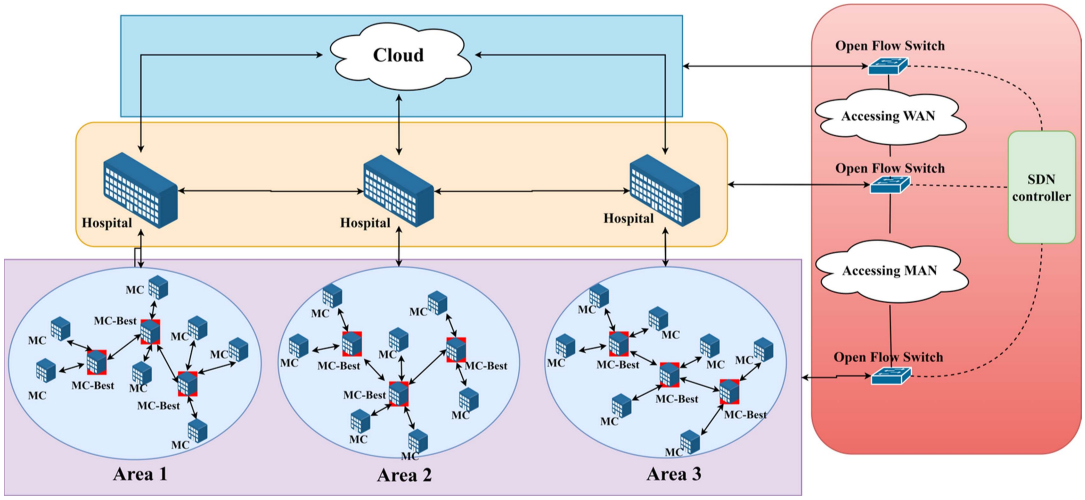


Fig. 1. Proposed edge/fog healthcare architecture based on SDN technology.

context of a software-defined cloud-fog network. Their algorithm improves performance by leveraging the opposite property of mutated particles and reducing the inertia weight linearly. Dong et al. [21] proposed HEELS, a task deployment strategy for load balancing in edge computing combined with cloud computing. By leveraging clustering analysis and an improved GSO algorithm with SCA, they achieved efficient task deployment and long-term load balancing. This study aligns with the goal of optimizing resource allocation and performance in edge computing environments.

Furthermore, Shahrbabaki et al. [22] proposed an SDN-enabled scheme for load balancing in edge nodes used for real-time IoT video data analytics. By utilizing incoming and outgoing traffic load measurements, the scheme estimates the load at each server and assigns a load score to determine workload distribution. Through periodic rerouting of IoT video streaming flows based on load scores, the proposed solution achieves a balanced workload distribution. The effectiveness of this scheme is demonstrated through simulation results, showcasing its potential in reducing the average video frame data analytics at edge nodes through workload balancing. Chen et al. [23] proposed a novel method called TDBEC for load balancing in multi-edge collaboration in WMANs. TDBEC employs a two-stage decision-making approach using DNN-based and DQN-based models to optimize task scheduling and adjust operations. The method achieves load balancing through centralized and decentralized decision-making based on global and local information, respectively. By incorporating these decision-making models, TDBEC provides an efficient load-balancing mechanism for multiedge collaboration in WMANs.

In summary, the existing methods in the literature can be mainly categorized into static, dynamic, and occasionally centralized or/and decentralized load-balancing techniques. The majority of existing research in this field has primarily focused on addressing the load balancing problem through separate utilization of these methods. However, this approach has revealed limitations in terms of adaptability, real-time responsiveness, and the requirement of significant resources. In contrast to previous research, the proposed algorithm integrates these diverse approaches, combining the simplicity of static methods with the adaptability and responsiveness of dynamic methods while utilizing fewer resources. Moreover, it incorporates localized decision-making centralization and network collaboration or decentralization across different areas. The proposed approach employs a load-balancing algorithm to make initial load-balancing decisions and subsequently utilizes an SDN-based algorithm to offload data within the network, either locally within the areas or between them. This combined approach offers a practical and scalable solution to the load-balancing challenge in edge/fog-based healthcare systems. To the best of our knowledge, this study is the first to propose such a combined static and SDN-based load-balancing method specifically for edge/fog-based healthcare applications.

III. Proposed System Model

Fig. 1 presents a conceptual framework for healthcare systems in urban areas that leverages edge computing and SDN technology. This model embodies a modern approach to healthcare, utilizing edge computing and SDN technology to deliver fast, efficient, and reliable healthcare services to patients. The framework comprises various components, including Medical Centers (MCs), Hospitals (Hs), and Cloud, in conjunction with patients located throughout the entire region.

SDN technology is used to manage the network and make intelligent decisions about how and where to process data. This is facilitated through a centralized SDN controller, which maintains a global view of the network and can make informed decisions about data flow.

The urban environment is conceptualized as a 3-D grid that comprises three distinct entities, namely, Hs, MCs, and Patients (users). Each H is responsible for providing medical services to a group of MCs located within a particular geographic region. Similarly, each MC is responsible for providing medical services to patients within its local vicinity. Patients, on the other hand, are dispersed throughout the entire region.

To simplify system representation, a set of assumptions has been established. First, we assume that every MC situated within the grid can potentially serve as a suitable candidate for the placement of edge-servers. Second, we postulate that all MCs are connected in a virtual sense only when the distance between any two MCs, as computed by the distance function *d(a,b),* is equal to a predetermined value *K*. This *K* value represents the maximum distance that can be utilized for establishing connectivity between neighboring MCs. Finally, we assume that each MC has a distinct historical load, i.e., a unique volume and distribution of user requests over time.

Consequently, the collection of MCs in the system is denoted as MCs = $\{\text{mc}_1, \text{mc}_2, ...; \text{mc}_n\}$, where each mc denotes a feasible placement location within the 3-D grid, and noMCs represents the number of MCs within that specific region. The set of ESs required for complete coverage in this area is represented as ES = $(\text{es}_1, \text{es}_2, ...; \text{es}_k)$, with Nes denoting the total number of ESs necessary. It is important to note that all ESs are considered identical (i.e., homogeneous). The patient population is denoted by $P = (p_1, p_2, \ldots; p_m)$, where *m* corresponds to the total number of patients in the region.

IV. Problem Formulation

According to [24], the MCs in the system can be classified into two distinct categories, namely MCs-Best and MCs-Others. MCs-Best are identified as the optimal placement sites for ESs, with the constraint that at least three MCs must be designated as such to enable efficient data offloading to neighboring MCs, specifically, the MCs directly connected with it. Conversely, MCs-Others are designated as serverless nodes that must be directly connected to MCs-Best to prevent multihop transfers and minimize latency. The tasks in the system can be accomplished through one of two methods. The first method involves offloading and processing tasks within the local network, irrespective of load balance. However, if the computational resources of the existing network are insufficient, the data may be processed in the cloud through offloading operations, as noted in [25]. It is worth noting that the increased response latency observed in the system when the number of tasks increases is attributed to the fact that the proposed algorithms did not take into account load balance. Load balancing is an important factor that affects the performance of distributed systems, and it is essential for achieving efficient utilization of resources and minimizing response time. Therefore, this study should consider incorporating load-balancing mechanisms into the algorithms to improve system performance and reduce response latency.

To address these problems, two main approaches have been proposed. First, the utilization of SDN technology has been suggested, which involves granting decision-making authority to the SDN controller in each region. This approach allows for optimal offloading decisions to be made for tasks, taking into account a global view of the network and compute resource allocation. By leveraging SDN technology in this way, the proposed approach aims to reduce response latency and improve system performance. Second, the proposed approach involves improving the OESP algorithm for selecting the best server sites and reducing them if possible. This approach aims to minimize the number of MCs designated as MCs-Best, while still ensuring full coverage for all MCs-Others. By optimizing the placement of MCs-Best, the proposed algorithm aims to reduce deployment costs while also enhancing the network's load balancing capabilities.

By combining these two approaches, the proposed solution aims to enhance the performance of edge-based healthcare networks, improve load balancing, and reduce costs. This approach has the potential to make a valuable contribution to the field of edge computing research, as it addresses some of the most significant challenges facing the development and deployment of edge computing networks.

To achieve these tasks, a multiobjective optimization model has been developed, with the following decision variables defined.

1) X represents the placement of edge-servers at MCs

$$X = \{x_j \mid 1 \le j \le n\}$$

where

$$x_j = \begin{cases} 1, & \text{if } \text{es}_i \text{ placed at } \text{mc}_j \\ 0, & \text{otherwise} \end{cases}$$

2) Y represents the assignment of patients to MCs

$$Y = \{y_{ij} \mid 1 \le i \le m,\ 1 \le j \le n\}$$

where

$$y_{ij} = \begin{cases} 1, & \text{if } p_i \text{ is assigned to } \text{mc}_j \\ 0, & \text{otherwise} \end{cases}$$

3) E represents the links between MCs

$$E = \{e_{ab} \mid,\ 1 \le a,\ b \le n,\ a \ne b\}$$

where

$$e_{ab} = \begin{cases} 1, & \text{if } \text{mc}_a \text{ and } \text{mc}_b \text{ are directly connected} \\ 0, & \text{otherwise} \end{cases}.$$

Let λ_{sum} be the set of task arrival rate of patients λ_i

$$\lambda_{\text{sum}} = \sum_{1=1}^{m} \lambda_i. \tag{1}$$

The system response time (SRT) represents the average system latency required to deliver patient data to a health facility within the architecture [25]. It is calculated as follows:

$$\text{SRT} = \frac{\sum_{i=1}^{n} t_{pi}}{n} \tag{2}$$

where

$$t_{pi} = T_{wi} + E_{\text{AP}} * T_{\text{AP}} + (c_1 + 1) * t_{\text{MC}} + c_1 * V + c_2 * S + c_2 * t_H + c_3 * L + c_3 * t_H + c_4 * B + c_4 * t_{\text{Cloud}} \tag{3}$$

$$t_{\text{MC}} = f_Q\ (K_1.\lambda_{\text{MCsum}}) + \frac{K_1.\lambda_{\text{MCsum}}}{\mu_{\text{MC}}} \tag{4}$$

$$t_H = f_Q\ (K_2.\boldsymbol{\lambda}_{H\text{sum}}) + \frac{K_2.\boldsymbol{\lambda}_{H\text{sum}}}{\mu_H}. \tag{5}$$

The t_{pi} is the average latency of the offloaded tasks by a patient, t_{MC}, and t_H are the latency at the MCs and Hs respectively. The *c1, c2, c3*, and *c4* represent the counters to count the number of times to reach a certain unit in the system.

The objective of this study is to improve the proposed architecture in [24] and [25] for all tasks generated by patients within the network while considering the delay requirements of task execution in healthcare systems. The optimization problem can be formulated as follows: given a system model with parameters (*G,* noMCs [points], *K,* Nes, *m,* T_{wi}, T_{AP}, λMAX), the goal is to find *X* among the best MCs and balance the system to enhance the response time SRT as described in [24] and [25]

$$\min \sum \mathrm{Cost}\,(\mathrm{esj},\ \mathrm{mck})\,.\,Xj \tag{6}$$

$$\min \mathrm{SRT} \tag{7}$$

Subject to

$d(\mathrm{mc}_a, \mathrm{mc}_b) = K$ *:* to ensure a shorter distance between a patient and an MC and to avoid the colocated problem.

$\sum E_{ij} = 1$ or 2 : to ensure that a patient either connects directly with an edge-servers site or through only one MC to minimize the latency.

$\sum y_{ij} = 1\ (1 \le j \le n)$: to guarantee that all patients are served, and each patient should be served from exactly one candidate MC.

By formulating the problem in this way and applying the proposed algorithm, the response time and load balancing in the healthcare system can be improved. The results of this study are expected to make a valuable contribution to the field of healthcare systems and edge computing research.

The solution to the optimization problem involves two algorithms that collaborate to tackle the challenges of diminishing edge computing expenses and minimizing network latency. Algorithm 1 focuses on reducing the number of ESs, thereby minimizing the associated cost, while simultaneously executing load-balancing tasks on the chosen nodes. Building upon this, Algorithm 2 utilizes a greedy heuristic method to manage data flow across the network, ensuring balance in load distribution and optimal resource utilization, thereby further diminishing latency. By integrating both algorithms, this solution presents a well-rounded approach, effectively addressing cost management, latency reduction, and resource allocation optimization through proficient load balancing.

V. Proposed Load Balancing Algorithms

The key objective of the previous study [24] was to effectively deploy ESs to MCs within a region to suit the demands of all monitored patients. The study employed the OESP algorithm, which demonstrated success in identifying the optimal placement of ESs in MCs-Best. The number of MCs-Best selected was determined by the shape and size of the network. Each MC-Best was tasked with providing services to several MCs-Others (i.e., serverless) based on specific parameters defined by the OESP algorithm. Consequently, the proposed algorithm succeeded in identifying a cost-effective network with reduced latency but exhibited an imbalance in workload distribution.

Algorithm 1: Load Balancing of Optimal Edge-Servers Placement (LB-OESP) Algorithm.

Input: A map of MCs-Best and MCs-Others, K = 40 km **Output:** Optimise the Cost (4) and SRT (5)	
1	Combine all MCs-Best in a one matrix called MCs_Best. Combine all MCs-Others in a one matrix called MCs_Other. **for** each MC_Best, **do** Plot a circle with a radius of K = 40 km. **End**
2	Generate an MCs_Best_details matrix = 0. **for** each MC-Best, **do** Find MC_Best_Group matrix. MCs_Best_details = MCs_Best_details + MC_Best_Group. **End**
3	**for** each MC-Best, **do** if MC_Best_Groupi ⊆ MC_Best_Groupj **Delete** MC_Best from MCs-Best. **Add** MC_Best to New-Other. **end** Best-MCs-Best = updated (MCs_Best). MCs-Others = MCs_Other + New-Other.

4	Generate ConnectivityBest = 0;
	for i ← 1 to size-of(Best-MCs-Best), **do**
	for k ← 1 to size-of(Best-MCs-Best), **do**
	if i ~= k, **then**
	if distance between (Best-MCs-Best)i and (Best-MCs-Best)j <= 40, **do**
	ConnectivityBest(i,k) = 1;
	Connected_MCs = Connected_MCs + (Best-MCs-Best)i.
	else
	ConnectivityBest(i,k) = 0.
	Not_Connected_MCs = Not_Connected_MCs + (Best-MCs-Best)i
	end
	else
	ConnectivityBest(i,k) = Null,
	end
	end
	end
	if Not_Connected_MCs == Null, **then**
	Go to 7
	else
	Go to 5
	end

5	**for** each Not_Connected_MCs, **do**
	if distance between Not_Connected_MCs and New-Other <= 40, **then**
	Add New-Other to Best-MCs-Best
	end
	end

6	Repeat 4 to test connectivity again.
7	Re-plot the MCs position in the map.

8	Find new connections between the MCs-Best
	for i ← 1 to size-of(Best-MCs-Best), **do**
	for k ← 1 to size-of(Best-MCs-Best), **do**
	if i ~= k, **then**
	Find the nearest Best-MCs-Best.
	If the nearest MC connection < 3
	Connect this MC to (Best-MCs-Best)i.
	end
	end
	end
	end
	Plot the new links between Best-MCs-Best.

9	Find the number of MCs-Others that should connected to each MC-Best with load balancing.
	No_Other_BestMC = No_Other/No_Best.
	R1 = int(No_Other_BestMC). % The minimum number
	R = R1 - No_Other_BestMC.
	if R == 0, **then**
	R2 = R1.
	else
	R2 = R1 + 1. % The maximum number
	end
	Distribute R1 or R2 of MCs-Others to MCs-Best to achieve the best load balancing.

10	Plot the final map based on each found connection between the Best and Other,

The primary aim of this article is to propose a novel load-balancing technique that can effectively address the issue of load imbalance in similar networks. The main steps involved in designing such an algorithm can be summarized as follows.

Step 1: Apply the OESP algorithm [24] to identify the optimal sites (MCs-Best) for placing ESs among several other sites (MCs-Others) in a specific area.

Step 2: Implement a static load-balancing algorithm to reorganize and manage the connections between the MCs within the local area. This task involves linking MCs-Others to MCs-Best in a manner that balances the loads among MCs-Best. The technique is based on the number of sites and historical data on facility loads. To achieve this, the MCs-Others should be distributed as evenly as possible among the MCs-Best. The proposed algorithm for accomplishing this is called LB-OESP and can be summarized in the following stages.

1) Define the maximum coverage of each MC-Best to gain a better understanding of the network.
2) Determine the MCs-Best details matrix, which includes each MC-Best and its corresponding set of MCs-Others.
3) Identify the Best-MCs-Best (the most optimal MCs-Best sites) to eliminate any redundant or substitutable MCs-Bests.
4) Test Connectivity 1) between the new MCs-Best to determine if they can be connected or if additional MCs-Best needs to be added.
5) If the result of 4) is false, identify New-MCs-Best that need to be added to the MCs-Bests to achieve complete connectivity between them.
6) Test Connectivity 2) between the final best sites and update the MCs-Best.

Algorithm 2: SDN-GH Algorithm for Data Offloading in Healthcare Architecture.

	Input: The final map of the architecture (MCs, H,, SDN, Cloud) **Output:** Optimise the SRT and load balancing
1	The respective SDN controller receives periodic state reports from each MC and H.
2	**for** each MC at a certain time, the SDN **do** Calculate C_{MC}(capacity) of each MC. Calculate C_H(capacity) of the local H. **end**

3	**for** each λ_{sum} arrived at the same time, the SDN **do**
	if $\lambda_{sum} < C_{MC}$ **then**
	Execute λ_{sum} at the local MC
	Terminate the process.
	Record the response time.
	else
	Execute C_{MC} of data.
	Calculate $\lambda_{off} = \lambda_{sum} - C_{MC}$.
	Record the response time.
	Go to 4.
	end
	end

4	**for** each λ_{off} arrived at the same time, the SDN **do** Calculate $t_{offloading}$. Calculate t_{local}. **end**

5	According to Equation 8:
	set the value of T_x^{xi}between the local MCs (x) and MCs neighbour x$i = \{x1, x2\}$ and the value of T_{x1}^Hbetween the local MCs ($x1$) and the local H.
	if T_x^{xi} and T_x^H == zero **then**
	Execute λ_{off} at the local MC.
	Record the response time.
	Terminate the process.
	else
	Construct $\beta_1 = \{T_x^{x1}, T_x^{x2}, T_{x1}^H\}$.
	end

6	**If** $\beta \neq \emptyset$ **then**
	Determine the MC or H index k = min β_l.
	Offload λ_{off} to k for the execution.
	The steps from 1-6 are applied to H with its neighbours and the cloud.
	else
	Execute λ_{off} at the local MC.
	Record the response time.
	Terminate the process.
	end

7	**Return** the total response time**.**

7) Replot all points without connections to distinguish the MCs-Best from the other sites.
8) Establish efficient connections between the MCs-Best to ensure full connectivity with the least number of links.
9) Determine and distribute the MCs-Others that should be connected to each MC-Best to achieve load balancing.

10) Identify the final MCs-Best details matrix, which includes each MC-Best and its corresponding set of MCs-Others.

Therefore, the proposed LB-OESP algorithm is a comprehensive approach that combines the OESP and static load-balancing algorithms to effectively balance the loads among the MCs-Best and MCs-Others, thus addressing the issue of load imbalance in similar networks. The LB-OESP algorithm is presented in Algorithm 1.

Step 3: To address the issue of resource scarcity within the local area, a dynamic load balancer is required. One potential solution to this problem is to employ a dynamic offloading strategy based on SDN. This approach considers the real-time status of the network to identify the most suitable node for offloading healthcare-related tasks. The SDN controller is responsible for making the necessary task assignment decisions required for offloading these tasks. Implementing this approach is expected to improve the utilization of available resources significantly, resulting in enhanced processing capabilities and optimal network utilization.

The proposed algorithm should be simple enough to allow for rapid decision-making while still meeting the delay requirements. However, it is unnecessary to offload a task to another lightly loaded site if the offloading time is greater than the local execution time. In certain scenarios, the time required for communication and remote execution may diminish the benefits of offloading to a less burdened site. To address this, we can define a binary decision variable T_{x1}^{x2} as

$$T_{x1}^{x2} = \begin{cases} 1, & \text{if } t_{\text{offloading}} < t_{\text{local}} \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

The primary function of the SDN controller is to collect data on the MCs, such as their current workloads, queue durations, and other relevant metrics. Having a global view of the network allows the controller to leverage AI-based technologies trained on offline data. This enables the controller to make more informed predictions, including average propagation delays, queueing delays, and result delivery times. Such forecasts can be valuable in assessing a task's response time if it were executed on a different site than its local site. To optimize SRT and load balancing, Algorithm 2 proposes an SDN-GH algorithm. This algorithm aims to determine whether data offloading is necessary between the system units based on the workload received by an MC or H relative to their maximum processing capacity. The algorithm also takes into account the latency time, which is calculated using (8).

The processing of the data follows a series of scenarios based on the processing capabilities of various components of the system. If the local MC-Best has enough processing power, the data is processed locally (Scenario 1). If not, the data is partially processed at the local MC-Best, and the remainder is divided to be offloaded to the neighboring MCs-Best according to their available capacities (Scenarios 2 and 3). If the neighboring MCs-Best do not have enough processing power, the data is partially processed at those sites, and the remaining portion is offloaded to the local H (Scenario 4). If the local H does not have sufficient processing power, the data is partially processed at that site, and the remaining portion is offloaded to the neighboring Hs, specifically, the Hs directly connected with it (Scenarios 5 and 6). If the neighboring Hs also lack adequate processing power, the data is partially processed at those sites, and the remaining portion is offloaded to the cloud (Scenario 7). Fig. 2 visualizes the seven scenarios.

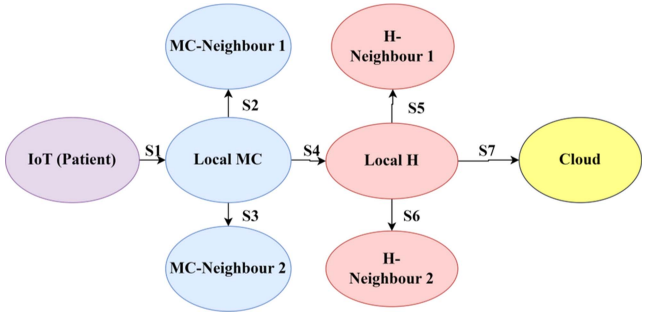


Fig. 2. All potential scenarios based on SDN-GH.

It is worth noting that in the previous study [25], the implementation of the HOSSC algorithm led to the successful achievement of five data processing scenarios, characterized by enhanced privacy, elevated computational capability, minimized system latency, sustained high service availability, and a scalable system design. In this study, we expanded the number of data processing scenarios to seven, which enabled us to improve upon the previously established outcomes while ensuring load balance and reducing delay time.

VI. Performance Evaluation

The performance of the proposed algorithms was evaluated using a simulator that combines SDN technology with edge computing for task offloading in the proposed system. The simulation was conducted on a notebook featuring a 1.8 GHz Intel Core i7-8550U CPU, 16 GB RAM, Microsoft Windows 10 operating system, and MATLAB R2022b. Table I lists the simulation parameters, primarily derived from our previous studies [24] and [25] . In addition, the following assumptions were made.

1) The arrangement of MCs is assumed to be random in terms of their distribution.
2) The topology is assumed to be known to the SDN controller, which has access to information regarding the distances between any two MCs in the front-haul networks.
3) Valuable information is presumed to be accessible through the utilization of AI-based learning technologies. Hence, parameters like the average propagation delay between MCs are predetermined.

In evaluating performance, the proposed algorithms are compared with those introduced in our previous studies to demonstrate the improvements achieved through the integration of new technologies into our foundational system. Subsequent sections present the results of implementing the proposed algorithms.

TABLE I
SRT SIMULATION PARAMETERS

Symbol	Parameter	Value
Ps	Number of patients	100, 200, 300
n	Number of servers in each MC	5, 6, 7
m	Number of servers in each H	10, 12, 14
r_{Wi}	Link rate between IoT device and AP	54 Mbps
r_{AP}	link rate between AP and the local MC	100 Mbps
r_{Fib}	link rate between the MCs and Hs	Up to 10 Gbps
λi	Packet size	30 KB
T_{wi}	Wireless Delay	4 ms
T_{AP}	Delay between AP and the local MC	2 ms
V	Delay between two neighboring MCs	$(1-K_1).\boldsymbol{\lambda}_{MCsum}/10$ Gbps
S	Delay between the MCs and Hs	$K_2.\boldsymbol{\lambda}_{Hsum}/10$ Gbps
L	Delay between two neighboring Hs	$(1-K_2).\boldsymbol{\lambda}_{Hsum}/10$ Gbps
B	Internet Delay	20 ms
μ_{MC}/ μ_H	Each MC/H server service rate	100/200 KB per ms
μ_{Cloud}	Cloud service rate	1000 KB per ms
λ_{MCmax}	Maximum MC workload	200×n KB
λ_{Hmax}	Maximum H workload	200×m KB

TABLE II
LB-OESP SIMULATION PARAMETERS

Symbol	Definition	value
A	Region dimensions	100 km * 100 km
noMCs	Number of MCs	12, 16, 25
x, y	Coordinates	Randomly created
Dmax	Max. distance between two neighbors	40 km

A. LB-OESP Algorithm

In this section, the results of multiple simulations are presented to assess the effectiveness of the proposed algorithm. The evaluations were based on randomly generated MC sites, depicted as points. Since the output of the OESP algorithm [24] serves as the input for the LB-OESP algorithm, identical parameters are defined for both algorithms, as detailed in Table II. To demonstrate the effectiveness of the approach, the step-by-step process of the algorithm is showcased initially with ten randomly generated, followed by the presentation of final outcomes when applying the algorithm to 20 and 30 randomly generated points.

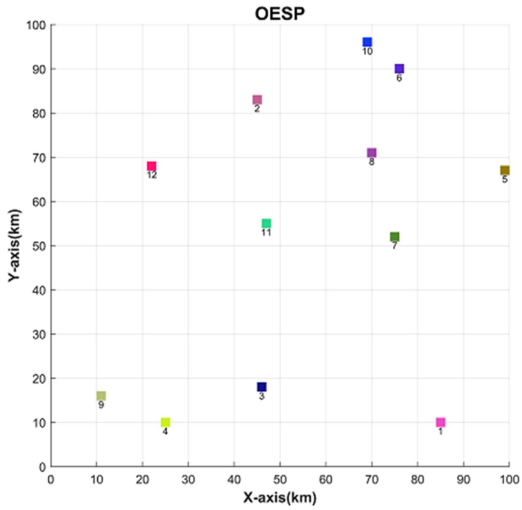


Fig. 3. 10 disconnected MCs on a map.

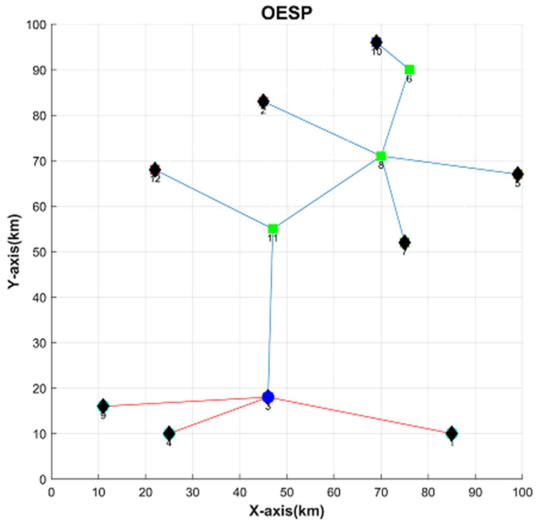


Fig. 4. Final OESP network of 10 MCs: Green & Blue points = best-MCs, Black points = MCs-others.

1) Scenario 1–12 Nodes: Initially, a MATLAB code was used to generate ten points within a geographic region measuring 100 km by 100 km. These points were designated as measurement sites (MC sites), as previously noted. Subsequently, it was necessary to allocate these points on a virtual map, as depicted in Fig. 3.

The next step involves using the OESP algorithm to determine the placement of MCs-Best and MCs-Others, which serve as inputs to the proposed algorithm. The final map generated by the OESP algorithm is illustrated in Fig. 4. It is evident that the OESP algorithm successfully identified optimal locations (MCs-Best), specifically (6, 11, 8, and 3), for server placement. However, concurrently, it overlooked considerations for load balancing by not taking into account the fair distribution of MCs-Others equally among MCs-Best sites. For instance, certain MCs-Best are solely connected to one MCs-Others, such as 6 and 11, while others are linked to three MCs-Others, such as 8 and 3.

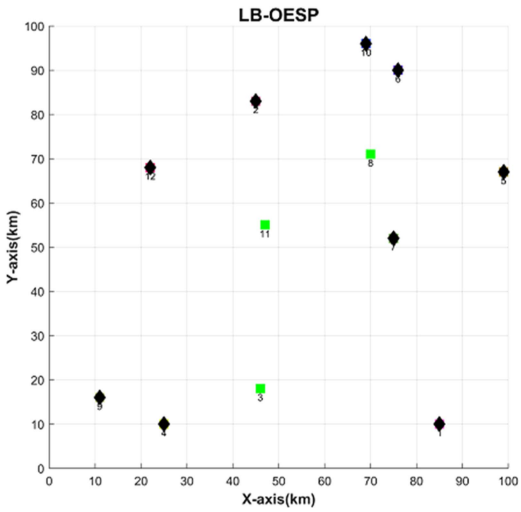


Fig. 5. Ten disconnected MCs on a map. Green points = best-MCs-best, Black points = MCs-others.

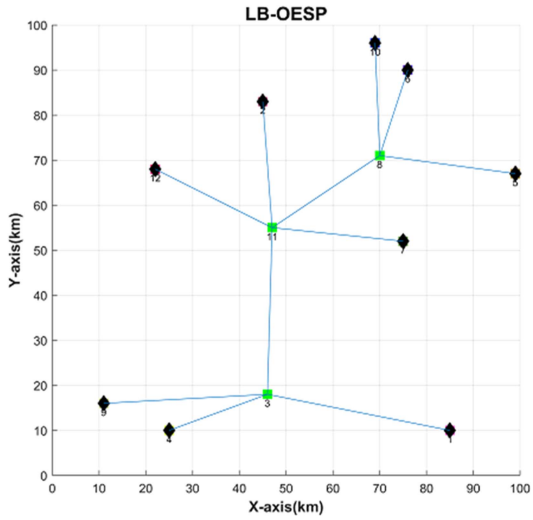


Fig. 7. Final LB-OESP network of ten MCs: Green points = best-MCs-best, Black points = MCs-others.

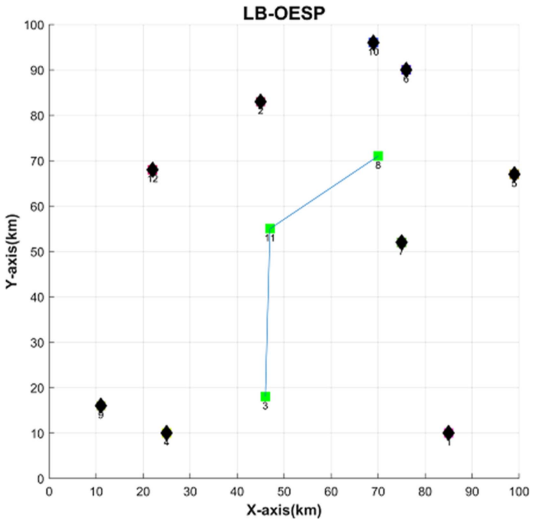


Fig. 6. Established connections between the best sites in the network.

Consequently, the central aim of the algorithm is to achieve a balanced distribution of load in the network. Prior to this objective, the proposed algorithm seeks to enhance the network by selecting the optimal MCs-Best to reduce redundancies and minimize costs. To accomplish this, the first task involves determining the Best-MCs-Best through the evaluation of multiple variables and the implementation of several procedures as detailed in steps 1–6 of the algorithm. Once completed, the network will be reconfigured, with connections between sites being adjusted, as illustrated in Fig. 5.

As depicted in Fig. 5, the selection of the Best-MCs-Best was limited to three sites, which are (8, 11, and 3), rather than four. MC 6 has been removed from MCs-Best and assigned to MCs-Others. The subsequent task is to establish optimal connections between these sites while avoiding duplicated links, accomplished through step 8 of the algorithm. Fig. 6 illustrates the configuration of the network with the connections established between the best sites.

Once the MCs-Best were established and connections between them were in place, the next step involved integrating the MC-Others into the network. The objective at this stage was to determine an optimal method of distributing the MC-Others as evenly as possible among the best sites. This involved determining the preferred number of MC-Others to be connected to each MC-Best, considering the minimum and maximum threshold. In this particular scenario, with nine MC-Others and three MCs-Best, the optimal number was three MC-Others (both as the minimum and maximum) to be connected to each MC-Best, taking into account the proximity of the interconnected sites. Fig. 7 depicts the final configuration of the grid after the implementation of steps 9 and 10. The algorithm has successfully established a network that is both balanced and cost-efficient in terms of its selection of sites to serve as servers for other sites.

It is important to note that when the preferred number of MC-Others is not a whole number, the approach should be to select the nearest integer values. For example, if the preferred number is 3.3, the minimum value would be 3 and the maximum would be 4. The algorithm then determines which MC-Best will accommodate 3 or 4 MC-Others based on the proximity of the MC-Others to the respective MCs-Best.

Furthermore, the selection of the MCs-Best is conducted in two stages. The first stage involves selecting the Best-MCs-Best1 from the MCs-Best and isolating the remaining elements in a separate matrix. Subsequently, the connectivity between the Best-MCs-Best1 is tested. If they are connected, the Best-MCs-Best1 are considered the Best-MCs-Best. In case of disconnection, the algorithm proceeds by adding one MC at a time from the isolated matrix until full connectivity between them is achieved.

2) Scenario (2 and 3)—16 and 25 Nodes: The effectiveness of the proposed algorithm is further highlighted in Figs. 8 and 9, which illustrate two distinct scenarios involving 16 and 25 nodes, respectively. These figures showcase the output maps generated

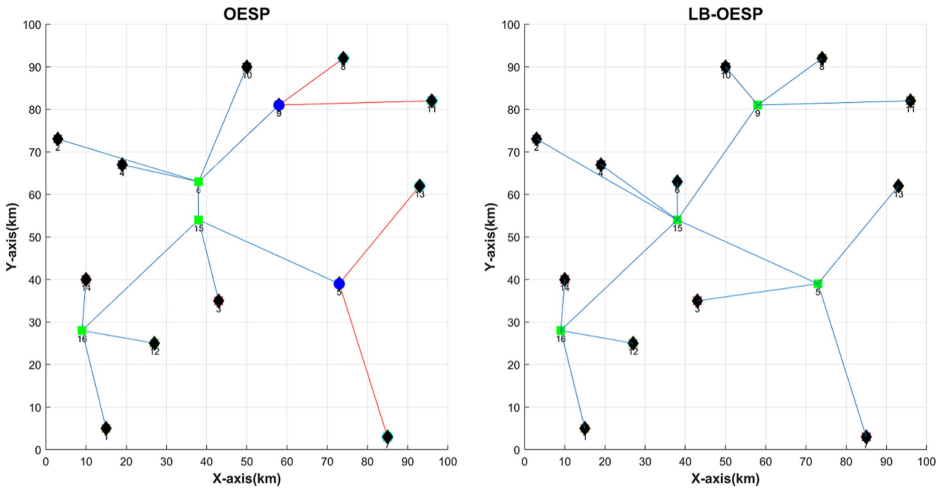


Fig. 8. Comparison between the final OESP and LB-OESP network of 16 MCs.

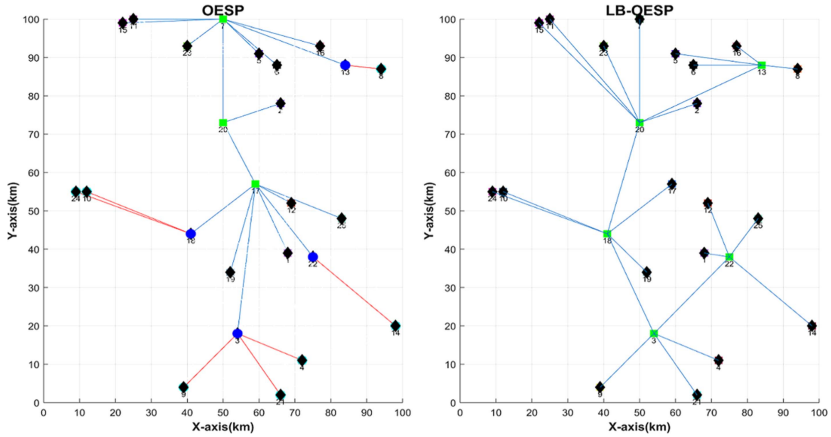


Fig. 9. Comparison between the final OESP and LB-OESP network of 25 MCs.

by both the OESP algorithm and the subsequent LB-OESP algorithm. The algorithm consistently demonstrates its ability to minimize the number of MCs-Best while achieving optimal load balancing. It successfully restructures the network by establishing connections between MCs-Best and linking MCs-Others to MCs-Best in an optimal manner, thereby enhancing overall network efficiency and performance.

In the 16-node scenario, the proposed algorithm successfully decreased the number of MCs-Best from 5 to 4 and then reconfigured their connections. Subsequently, it evenly distributed the MCs-Others among the MCs-Best.

In the 25-node scenario, while the LB-OESP algorithm successfully reduced the number of MCs-Best and reconfigured the network, it was necessary to connect MC-Best 20 to a larger number of MCs-Others than MC-Best 3. This was due to the remote location of MC-Best 3, limiting its ability to connect to more MCs-Others. This is a prime example of the algorithm's positive aspects, as it possesses the intelligence necessary to construct a network that is optimally connected, incurs the lowest cost, and experiences the shortest possible delay time.

TABLE III
COMPARISON BETWEEN HMAN AND OESP COST

noMCs	Number of ES nodes in			Deployment Cost
	HMAN [25]	OESP [24]	LB-OESP	
12	12	4	3	-25%
16	16	5	4	-20%
25	25	7	5	-28%

Table III provides a cost comparison between the proposed algorithm and previous studies [24], [25], highlighting the differences. It is important to note that these percentage variations may differ depending on the specific network configurations, as they are influenced by the selection of the best points. The choice of best points is dependent on the network's shape and characteristics. Nonetheless, significant cost improvements are evident across all scenarios.

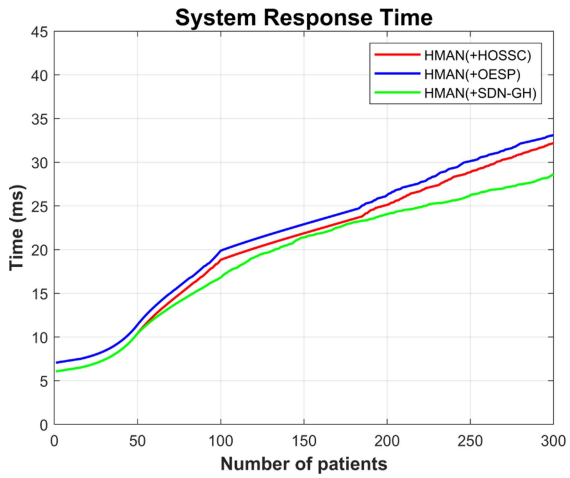


Fig. 10. Comparison between HOSSC, OESP, and SDN-GH algorithms in terms of the SRT in the system.

B. SDN-GH Algorithm

The SDN-GH algorithm has been designed to efficiently distribute workloads among the collaborating units, thereby alleviating resource constraints and ensuring high availability within the system. Accordingly, upon receipt of a request by a node within the local area, the system evaluates whether the resources on that node suffice for processing the request. In cases where these resources are inadequate, the algorithm transfers the request across the cooperating nodes where the necessary resources are available. If resources are unavailable across all units, the request is redirected either to another area or to the central cloud.

The results of the conducted simulation are depicted in Fig. 10, where the *y*-axis represents the delivery latency for task processing within the system's facilities and the *x*-axis signifies the number of patients. For the purpose of comparison, two additional algorithms, the OESP algorithm and the HOSSC algorithm as outlined in [24] and [25] are employed. To facilitate a more straightforward evaluation of the system's viability and benefits, several parameters have been defined and presented in Table I, which are identical to the parameters used in [24] and [25].

It is evident that the algorithm endeavors to minimize processing time by initially involving proximate collaborative facilities. Subsequently, it gradually extends its reach to facilities situated farther away, eventually culminating in engagement with the cloud if necessary. This operational approach aligns with the fundamental tenets of edge technology. During periods of low workloads, specifically when patient numbers are less than 100, the algorithm relies predominantly on the local MC (MC_L) and its neighboring centers (MC_{N1} and MC_{N2}). Consequently, the delay is kept to a minimum. As the workload increases, additional collaborative facilities are hierarchically included. This expansion encompasses the local hospital (H_L) and its neighboring hospitals (H_{N1} and H_{N2}), eventually extending to the cloud. The decision-making process is guided by the algorithm's adherence to the seven scenarios delineated in Fig. 2.

TABLE IV
NUMBER OF PATIENTS SERVED IN EACH UNIT (PURPLE INDICATES UNUSED UNITS AT A CERTAIN STAGE)

Ps	MC_L	MC_{N1}	MC_{N2}	H_L	H_{N1}	H_{N2}	Cloud
	Total Capacity (C_T) = 50			C_T = 100			C_T=inf
100	50	23	27				
200	50	41	33	42	17	17	
300	50	35	31	72	58	54	
400	50	39	32	69	59	62	89

Furthermore, the results indicate a significant enhancement in the system's performance when utilizing the SDN-GH algorithm compared to the HOSSC and OESP algorithms. This improvement can be attributed to the SDN-GH algorithm's broader scope of data processing within the network, offering six scenarios for local network data processing and only one scenario involving data transmission to the cloud, resulting in reduced response time. This confirms the efficacy of the proposed algorithm in promoting a more collaborative approach by distributing workloads among the system units. In addition, this reinforces the privacy and availability of the system while contributing to scalability and capacity improvements, as previously reported in [25]. This also marks a considerable enhancement when compared to the latency of 75 ms recorded in [26], which utilized almost identical simulation parameters. The noteworthy decrease in service delay, quantified at over 50%, is ascribed to the pioneering load-balanced architecture employing the nearest units for collaborative data processing. This comparison emphasizes the superior performance of our proposed framework in real-world healthcare scenarios.

Finally, to corroborate the algorithmic outcomes concerning efficiency and scalability, Table IV presents pertinent scalar findings derived from the system, affirming the efficacy of the algorithm in the prudent allocation of workloads to interconnected health facilities. This allocation is adaptively executed in accordance with the available capacity at each facility. The load distribution process is discernibly balanced to a degree that facilitates the system's operability under diverse loads, thereby affirming the scalability of the system, particularly after the implementation of the proposed load balancing algorithm.

VII. CONCLUSION

This study addressed the crucial challenge of load balancing in the design of smart healthcare systems. Efficient load balancing is essential to avoid overloading nodes, which can result in delays and degraded system performance. By integrating static and SDN-based load-balancing algorithms, the proposed method achieved optimal load balancing in edge/fog-based healthcare systems. The LB-OESP algorithm minimized the number of ES deployment locations while ensuring balanced workload distribution. The SDN-GH algorithm dynamically balanced the load, leading to improved system performance. The results showed a 12% decrease in system latency and up to 28% lower deployment costs compared to prior studies, highlighting the effectiveness of the proposed method. Furthermore, the implementation of the SDN-GH algorithm enabled seven data processing scenarios,

enhancing privacy, computational capability, system latency, service availability, and system design adaptability. These findings emphasize the significance of efficient load balancing in improving system performance in healthcare applications. Future research directions could explore the scalability of the proposed method for larger networks, investigate its resilience to network failures and security threats, and extend its applicability to other domains beyond healthcare. Overall, this study contributes to advancing the field of smart healthcare systems and provides valuable insights for designing efficient and reliable edge/fog-based healthcare networks.

REFERENCES

[1] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Trans. Serv. Comput.*, vol. 16, no. 2, pp. 1505–1521, Mar./Apr. 2023, doi: 10.1109/TSC.2022.3174475.

[2] S. Ebneyousef and A. Shirmarz, "A taxonomy of load balancing algorithms and approaches in fog computing: A survey," *Cluster Comput.*, vol. 26, no. 5, pp. 3187–3208, 2023, doi: 10.1007/s10586-023-03982-3.

[3] C. S. M. Babou et al., "Hierarchical load balancing and clustering technique for home edge computing," *IEEE Access*, vol. 8, pp. 127593–127607, 2020, doi: 10.1109/ACCESS.2020.3007944.

[4] M. Kyryk, N. Pleskanka, M. Pleskanka, and P. Nykonchuk, "Load balancing method in edge computing," in *Proc. IEEE 15th Int. Conf. Adv. Trends Radioelectron., Telecommun. Comput. Eng.*, 2020, pp. 978–981, doi: 10.1109/TCSET49122.2020.235584.

[5] H. Pydi and G. N. Iyer, "Analytical review and study on load balancing in edge computing platform," in *Proc. 4th Int. Conf. Comput. Methodol. Commun.*, 2020, pp. 180–187, doi: 10.1109/ICCMC48092.2020.ICCMC-00036.

[6] T. A. Al-Janabi and H. S. Al-Raweshidy, "Optimised clustering algorithm-based centralised architecture for load balancing in IoT network," in *Proc. Int. Symp. Wireless Commun. Syst.*, 2017, pp. 269–274, doi: 10.1109/ISWCS.2017.8108123.

[7] H. N. Al-Anbagi and I. Vertat, "Pre-detection combining of small satellite downlink's replicas," in *Proc. Int. Conf. Elect., Comput. Energy Technol.*, 2022, pp. 1–6, doi: 10.1109/ICECET55527.2022.9873068.

[8] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2359–2391, Oct.–Dec. 2017, doi: 10.1109/COMST.2017.2717482.

[9] C. Tang, C. Zhu, N. Zhang, M. Guizani, and J. J. P. C. Rodrigues, "SDN-assisted mobile edge computing for collaborative computation offloading in industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 24253–24263, Dec. 2022, doi: 10.1109/JIOT.2022.3190281.

[10] J. Li et al., "A secured framework for SDN-based edge computing in IoT-enabled healthcare system," *IEEE Access*, vol. 8, pp. 135479–135490, 2020, doi: 10.1109/ACCESS.2020.3011503.

[11] M. Priyadarsini, J. C. Mukherjee, P. Bera, S. Kumar, A. H. M. Jakaria, and M. A. Rahman, "An adaptive load balancing scheme for software-defined network controllers," *Comput. Netw.*, vol. 164, pp. 1–11, 2019.

[12] K. Amirthalingam, "Medical dispute resolution, patient safety and the doctor-patient relationship," *Singap. Med. J.*, vol. 58, no. 12, pp. 681–684, 2017, doi: 10.11622/smedj.2017073.

[13] I. Balansard et al., "Revised recommendations for health monitoring of non-human primate colonies," *FELASA Work. Group Rep. Lab. Animals*, vol. 53, no. 5, pp. 429–446, 2019, doi: 10.1177/0023677219 44541.

[14] Y. A. Chen, J. P. Walters, and S. P. Crago, "Load balancing for minimizing deadline misses and total runtime for connected car systems in fog computing," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. IEEE Int. Conf. Ubiquitous Comput. Commun.*, 2017, pp. 683–690, doi: 10.1109/ISPA/IUCC.2017.00107.

[15] X. Wang, Z. Ning, and L. Wang, "Offloading in Internet of Vehicles: A fog-enabled real-time traffic management system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4568–4578, Oct. 2018, doi: 10.1109/TII.2018.2816590.

[16] Z. Ning, X. Wang, J. J. P. C. Rodrigues, and F. Xia, "Joint computation offloading, power allocation, and channel assignment for 5G-enabled traffic management systems," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3058–3067, May 2019, doi: 10.1109/TII.2019.2892767.

[17] A. Cabrera, A. Acosta, F. Almeida, and V. Blanco, "A dynamic multi-objective approach for dynamic load balancing in heterogeneous systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 10, pp. 2421–2434, Oct. 2020, doi: 10.1109/TPDS.2020.2989869.

[18] G. Li et al., "A new load balancing strategy by task allocation in edge computing based on intermediary nodes," *J. Wireless Commun. Netw.*, vol. 3, pp. 1–10, 2020, doi: 10.1186/s13638-019-1624-9.

[19] M. Z. Nayyer et al., "LBRO: Load balancing for resource optimization in edge computing," *IEEE Access*, vol. 10, pp. 97439–97449, 2022, doi: 10.1109/ACCESS.2022.3205741.

[20] X. He, Z. Ren, C. Shi, and F. Jian, "A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles," *Chin. Commun.*, vol. 13, no. S2, pp. 145–154, 2016.

[21] Y. Dong, G. Xu, Y. Ding, X. Meng, and J. Zhao, "A 'joint-me' task deployment strategy for load balancing in edge computing," *IEEE Access*, vol. 7, pp. 99658–99669, 2019, doi: 10.1109/ACCESS.2019.2928582.

[22] P. P. Shahrbabaki, R. W. L. Coutinho, and Y. R. Shayan, "A novel SDN-enabled edge computing load balancing scheme for IoT video analytics," in *Proc. IEEE Glob. Commun. Conf.*, 2022, pp. 5025–5030, doi: 10.1109/GLOBECOM48099.2022.10000605.

[23] X. Chen, Z. Yao, Z. Chen, G. Min, X. Zheng, and C. Rong, "Load balancing for multi-edge collaboration in wireless metropolitan area networks: A two-stage decision-making approach," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17124–17136, Oct. 2023, doi: 10.1109/JIOT.2023.3272010.

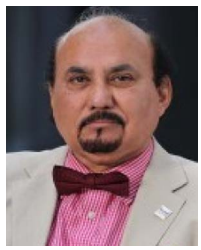
[24] A. M. Jasim and H. Al-Raweshidy, "Optimal intelligent edge-server placement in the healthcare field," *Inst. Eng. Technol. Netw.*, vol. 13, no. 1, pp. 13–27, 2023, doi: 10.1049/ntw2.12097.

[25] A. M. Jasim and H. Al-Raweshidy, "Towards a cooperative hierarchical healthcare architecture using the HMAN offloading scenarios and SRT calculation algorithm," *Inst. Eng. Technol. Netw.*, vol. 12, no. 1, pp. 9–26, 2023, doi: 10.1049/ntw2.12064.

[26] M. Asif-Ur-Rahman et al., "Toward a heterogeneous mist, fog, and cloud-based framework for the Internet of Healthcare Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4049–4062, Jun. 2019, doi: 10.1109/JIOT.2018.2876088.

Ahmed M. Jasim (Member, IEEE) received the B.E. degree in electronics and communications engineering from the University of Mosul, Mosul, Iraq, in 2007, the M.Sc. degree in computer systems and networks from East Ukrainian Volodymyr Dahl National University, Luhansk Oblast, Ukraine, in 2014. He is currently working toward the Ph.D. degree with Brunel University London, Uxbridge, U.K.

His permanent work is as a Lecturer with the Department of Computer Engineering, University of Diyala, Baqubah, Iraq. He has authored or coauthored a couple of journal and conference papers. His research interests include computer networks, IoT, and edge/fog/cloud computing.



Hamed Al-Raweshidy (Senior Member, IEEE) received the B.Eng. and M.Sc. degrees from the University of Technology, Baghdad, Iraq, in 1977 and 1980, respectively, the Postgraduate Diploma degree from Glasgow University, Glasgow, U.K., in 1987, and the Ph.D. degree from Strathclyde University, Glasgow, U.K., in 1991.

He was with the Space and Astronomy Research Centre, Baghdad, PerkinElmer, Waltham, MA, USA, British Telecom, London, U.K., Oxford University, Oxford, U.K., Manchester Metropolitan University, Manchester, U.K., and Kent University, Canterbury, U.K. He is currently the Director with the Wireless Network Communications Centre, Brunel University London, Uxbridge, U.K.