Innovative Methods for Edge Computing Deployment in Healthcare

Ahmed Mahmood Jasim
College of Engineering, Design and Physical Sciences
Brunel University London

A thesis submitted for the degree of
Doctor of Philosophy

2024

Acknowledgments

I am profoundly grateful to those who have supported me throughout this challenging yet immensely rewarding journey of pursuing my doctoral degree. The completion of this thesis would not have been possible without the encouragement, guidance, and assistance of numerous individuals and institutions.

I extend my heartfelt gratitude to my esteemed supervisor, Professor Hamed Al-Raweshidy, whose unwavering support, valuable insights, and scholarly guidance have been instrumental in shaping the direction of this research. Your dedication to academic excellence and commitment to my growth as a researcher have been a constant source of inspiration.

My gratitude extends to the Department of Electronic and Electrical Engineering/Brunel University London for providing the conducive research environment and resources that have been essential in carrying out this study. The collaborative atmosphere and diverse academic community have fostered an environment of growth and learning that I am truly thankful for.

I am indebted to my family (my parents, my wife, and my children) and friends for their unwavering encouragement, love, and patience. Your constant support, belief in my abilities, and understanding of the demands of this endeavor have been my pillar of strength.

Lastly, I would like to acknowledge all those whose contributions, whether big or small, have played a role in shaping this research. Your insights, discussions, and assistance have collectively contributed to the culmination of this work.

While it is not feasible to name everyone individually, please accept my sincere appreciation for your contributions to this academic journey.

Ahmed Jasim
2024

Abstract

This interdisciplinary research explores the integration of edge computing technology in the healthcare sector, presenting innovative methodologies across three key contributions.

In Chapter 3, the first contribution introduces the Healthcare Metropolitan Area Network (HMAN), a novel cooperative hierarchical Edge/Fog computing-based architecture for urban healthcare systems. HMAN offers offloading scenarios and the HOSSC algorithm, tailored for versatile data processing. Simulation results demonstrate its potential as a scalable and robust healthcare system, with efficient computing capacity and service availability. HMAN also ensures patient privacy through local data storage and processing, making it a practical solution for serving a large number of patients.

The second part of this research, expounded upon in Chapter 4, comprises dual facets: an AI-based priority mechanism to identify urgent cases, aimed at improving Quality of Service (QoS) and Quality of Experience (QoE) is proposed. Then, an optimal edge-servers placement (OESP) algorithm to obtain a cost-efficient architecture with lower delay and complete coverage is presented. Results show reduced patient latency based on urgency, prioritising critical cases. The OESP algorithm selects optimal deployment sites, achieving over 80% cost-efficiency improvement. In summary, the study enhances healthcare system performance, cost-effectiveness, and reduces latency.

The third part of this research, encapsulated in Chapter 5, introduces 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. LB-OESP efficiently balances workloads while minimising edge server requirements, improving system performance, and reducing 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 this approach provides adaptive load balancing, considering changing network conditions, resulting in improved system performance and reliability. Furthermore, it achieves a 12% reduction in system latency and up to 28% lower deployment costs compared to previous methods, offering a promising, efficient, and cost-effective solution for Edge/Fog-based healthcare systems.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Background and Motivation

In an ever-evolving world, healthcare holds a pivotal position as a cornerstone of human well-being and a paramount determinant of life quality. In the field of healthcare, resources can be categorised as consumable (like medical tools) and non-consumable (such as medical professionals). Making optimal use of non-consumable resources, which include valuable medical staff, is a crucial step towards enhancing healthcare systems [1]. As a result, the importance of healthcare systems has grown significantly in recent years, driven by the urgent need to improve service delivery, particularly in environments with limited resources.

In parallel, the continuous rise in the number of patients, evident in the day-to-day influx, has posed unprecedented challenges for public sectors responsible for providing effective healthcare services. This rise in demand has placed considerable pressure on traditional healthcare frameworks, hindering their capacity to keep up with society's growing needs. These challenges dramatically increased the need for fresh, innovative approaches to healthcare management that can enhance efficiency while also managing costs effectively. Therefore, there is an urgent need to find creative solutions that offer the required flexibility, scalability, and cost-effectiveness to cater to the constantly expanding patient population.

The field of providing electronic health (e-health) stems from the necessity to enhance healthcare sector management, optimise the utilisation of resource, and curtail costs while upholding quality. These systems are presented as a positioned as a pioneering technological paradigm shift with the potential to entirely reshape the medical field. With a focus on remote patient monitoring and supervision, these systems display substantial potential in tackling existing and forthcoming healthcare challenges. By delivering high-quality medical care at a limited expense, e-health systems offer a beacon of hope, alleviating the financial burdens weighing heavily on healthcare institutions and heralding improved patient outcomes.

The journey towards e-health has been made possible by groundbreaking advancements in computational intelligence, mobile communication technologies, and the Internet of Things (IoT). In this context, the integration of edge computing holds immense promise for the development of more efficient and effective healthcare systems. Edge computing refers to the practice of processing data closer to its source, minimising latency and enhancing real-time analysis. This approach has given rise to sophisticated systems capable of real-time monitoring and effective management of patient health. One of the primary advantages of leveraging edge computing in healthcare is its potential to significantly reduce latency in data processing and analysis. In critical situations, such as remote patient monitoring or real-time diagnostics, every millisecond counts. By processing data at the edge, closer to the patient or medical device, healthcare professionals can access timely insights, leading to quicker and more accurate decision-making.

Moreover, the vast amounts of data generated by wearable devices, medical sensors, and other health monitoring tools can overwhelm traditional data centres and networks. Edge computing offers a distributed and decentralised solution, distributing the data processing load and ensuring seamless communication between devices and central systems. This scalability not only enhances the overall performance of healthcare systems but also lays the foundation for the integration of other technologies like artificial intelligence, which rely heavily on rapid data analysis.

Furthermore, edge computing can enhance patient privacy and security. By processing sensitive health data locally, at the edge, the need for transmitting data to centralised servers is reduced, minimising the risk of data breaches and unauthorised access. This aspect is particularly crucial in healthcare, where maintaining patient confidentiality is paramount.

The integration of edge computing technology presents an exciting opportunity to usher in a new era of healthcare, characterised by real-time insights, enhanced patient care, and increased efficiency. By embracing this innovative approach, healthcare systems can navigate the complexities of the modern world while continuing to provide optimal care to individuals and communities alike.

1.2 Problem Statement

The escalating global population, projected to reach 9.7 billion individuals by 2050 [2], poses significant challenges, particularly in overcrowded urban areas, adversely impacting residents' quality of life. Inadequate healthcare management contributes to an alarming 2.6 million annual fatalities, a number on the rise, as reported by the World Health Organization (WHO) [3, 4]. The impending future necessitates immediate implementation of effective solutions to address the myriad challenges that cities will face. One critical predicament is the potential strain on urban health systems, and the healthcare industry struggles to deliver high-quality care to patients efficiently.

A notable challenge faced by the healthcare industry is the inefficiency of patient transfers between distinct levels of healthcare providers, causing delays in care, escalated expenses, and diminished patient outcomes. Moreover, the current lack of collaboration between different levels of healthcare providers leads to fragmented care, with each provider operating in isolation and focusing solely on their expertise.

To tackle these pressing issues, the emerging field of electronic healthcare (e-healthcare) provides a promising solution, leveraging the power of technology and human-device interaction facilitated by IoT, Edge, Fog, and Cloud technologies. However, integrating these technologies presents several technical challenges:

1. Scalability: Ensuring the healthcare system can handle increasing amounts of data and a growing number of connected devices without compromising performance.

2. Availability: Maintaining consistent and reliable access to healthcare services, particularly in the face of high demand and potential system failures.

3. Capacity: Providing sufficient processing power and storage to manage vast amounts of healthcare data generated by IoMT devices.

4. Latency: Minimising response times to ensure timely processing and delivery of critical healthcare data, especially in time-sensitive situations.

5. Privacy: Protecting sensitive patient information by ensuring data is stored and processed locally whenever possible, thereby reducing exposure to external threats.

To address these challenges, Edge/Fog computing paradigms have been introduced as complementary technologies to cloud computing. The primary goal of Edge/Fog computing is to shorten the distance between users and data processing centers, thereby improving response times, bandwidth, and privacy.

This research proposes the design of a transformative healthcare system that promotes cooperation among different levels of healthcare providers. The proposed architecture aims to reduce costs, and elevate the overall quality of care, positioning it as a viable alternative to traditional healthcare approaches. Through leveraging the rapid advancements in communication and information technologies, this research seeks to establish a more collaborative and technologically driven healthcare system capable of meeting the challenges posed by urban overcrowding and improving healthcare management.

1.3 Aim and Objectives

The aim of this research is to develop an advanced and innovative pervasive intelligent healthcare system/network that caters to local observation for patients in need. The primary aim is to address key challenges in healthcare, encompassing scalability, data privacy, service availability, low latency, intelligence, and cost-effectiveness. To achieve this aim, the research will pursue the following interconnected objectives:

Objective 1: Designing a Scalable System (Chapter 3)

- Investigate existing architectures and approaches in the context of healthcare systems.
- Develop a scalable framework capable of efficiently accommodating diverse workloads and adapting to varying demands.

Objective 2: Ensuring High Data Privacy Levels (Chapter 3)

- Design robust privacy-preserving mechanisms to ensure the secure handling and storage of sensitive healthcare data.

Objective 3: Ensuring High Service Availability (Chapter 3)

- Develop mechanisms for data replication, load balancing, and failover to prevent service disruptions and single points of failure.

Objective 4: Achieving Low Latency (Response Time) (Chapter 3,4, and 5)

- Develop novel approaches for data transmission and processing techniques to achieve minimal response time.

Objective 5: Designing an Intelligent System (Chapter 4)

- Develop an intelligent priority mechanism for enhancing the quality of service (QoS) and quality of experience (QoE) in healthcare systems based on real-time patient data and contextual information.

Objective 6: Creating a Cost-Effective System (Chapter 4 and 5)

- Explore cost-effective methods that optimise resource utilisation while ensuring the system's requirements are fulfilled.

Upon thorough validation and evaluation of the system performance through rigorous testing, benchmarking, and comparative analysis with existing solutions as verifiable evidence of achieving these objectives, the research aspires to make a substantial contribution to the advancement of a cutting-edge pervasive intelligent system/network, catering to local observation for patients. This innovative system seeks to effectively tackle critical challenges encompassing scalability, intelligence, data privacy, service availability, low latency, and cost-effectiveness.

1.4 Hardware and Software

1.4.1 Software:

The software component of the proposed pervasive smart system/network, which aims to achieve scalability, intelligence, data privacy, service availability, low latency, and cost-effectiveness, will be carefully selected and implemented to support the research objectives.

To simulate and model various scenarios, MATLAB software, a widely-used tool in research and industry, will be employed. The powerful capabilities of MATLAB enable efficient data analysis, algorithm development, and system simulation. Utilizing MATLAB, different scenarios can be simulated and tested, allowing for the evaluation of the system's performance under various conditions.

1.4.2 Hardware:

The hardware infrastructure supporting the pervasive smart system/network will be carefully selected and configured to effectively meet the research objectives. The primary development and simulation platform will be a Lenovo laptop running Windows 10 Home 64-bit, equipped with an Intel Core i7 processor and 16GB of RAM.

The chosen laptop provides a powerful computational environment suitable for running complex simulations, executing resource-intensive algorithms, and analyzing data. Its robust processing capabilities ensure efficient data analysis and real-time decision-making, facilitating low-latency response times within the system.

1.5 Summary of Contributions

1.5.1 HMAN Architecture

Proposing a cooperative hierarchical Edge/Fog computing-based architecture called Healthcare Metropolitan Area Network (HMAN) for urban healthcare systems. The architecture aims to enhance the reliability and robustness of electronic health services by addressing scalability, availability, capacity, latency, and privacy concerns. HMAN utilises existing infrastructure in cities, such as medical centres and hospitals, to connect patients to the healthcare system. The architecture introduces HMAN offloading scenarios and a specially designed algorithm called HMAN offloading scenarios and SRT calculation (HOSSC) to enable efficient offloading and processing within the network. Simulation results demonstrate that the designed architecture achieves a ubiquitous and scalable healthcare system with competitive performance in terms of computing capacity and service availability. The HMAN system exhibits low latency, responding to 1 to 300 patients simultaneously sending with a short response time ranging from 6.043 to 31.45 ms. Additionally, the proposed architecture ensures patient privacy by locally storing and processing data in anticipated scenarios. The HMAN architecture offers a viable solution for providing healthcare services to a large number of patients.

1.5.2 Optimal Placement Algorithm

Focusing on improving the efficiency of healthcare systems. This contribution proposes an enhancement to the HMAN architecture in two parts: an AI-based priority mechanism to identify urgent cases and improve QoS and QoE, and an optimal edge-server placement (OESP) algorithm to achieve a cost-efficient architecture with reduced latency and comprehensive coverage. The results demonstrate that the proposed priority mechanism algorithms reduce latency for patients with greater needs, while the OESP algorithm selects optimal sites for deploying edge servers, resulting in a cost-efficient system with over 80% improvement. The study introduces an improved healthcare system with enhanced performance, cost-effectiveness, and reduced latency.

1.5.3 Load Balancing Techniques

This contribution aims to further enhance the HMAN architecture by introducing additional improvements that address the existing challenges. Specifically, the goal is to improve the load balancing aspect of Edge/Fog-based healthcare systems by proposing an adaptive load balancing method that combines static and Software-Defined Networking (SDN)-based algorithms. The LB-OESP algorithm is introduced for static workload balancing, while the SDN-GH algorithm manages dynamic data flow within the network. The simulations demonstrate the effectiveness of the proposed adaptive load balancing method for Edge/Fog-based healthcare systems. The method takes into account changing network conditions, resulting in improved system performance and reliability. Notably, it achieves a significant 12% reduction in system latency and up to 28% lower deployment costs compared to previous studies. The proposed method offers a promising and cost-effective approach to workload management, real-time monitoring, and effective administration of health systems in the context of Edge/Fog-based healthcare environments.

1.6 Publications

The research findings of this thesis have been disseminated through the publication of three scholarly journal articles.

1. **Ahmed M. Jasim** and H. Al-Raweshidy, "Towards a cooperative hierarchical healthcare architecture using the HMAN offloading scenarios and SRT calculation algorithm," IET Networks, vol. 12, no. 1, pp. 9-26, 2023, doi: 10.1049/ntw2.12064.

2. **Ahmed M. Jasim** and H. Al-Raweshidy, "Optimal Intelligent Edge-Server Placement in the Healthcare Field," IET Networks, 2023, doi: 10.1049/ntw2.12097.

3. **Ahmed M. Jasim** and H. Al-Raweshidy, "An Adaptive SDN-Based Load Balancing Method for Edge/Fog-Based Real-Time Healthcare Systems", IEEE Systems Journal, 2024, doi: 10.1109/JSYST.2024.3402156.

1.7 Structure of the Thesis

The organisation of this thesis can be described as follows:

Chapter 2: Literature Review

In this chapter, an extensive review of relevant literature is presented. It explores the existing theories, concepts, and research studies related to the research topic. Gaps and limitations in the literature are identified, highlighting the need for the current research.

Chapter 3: Towards a Cooperative Hierarchical Healthcare Architecture using the HOSSC Algorithm

This chapter introduces the initial contribution of this research, presenting a novel healthcare architecture based on Edge/Fog computing within urban areas. The proposed architecture, referred to as HMAN, is supported by a specialised algorithm known as HOSSC, designed to manage the data flow processes within it. The chapter ends with a presentation and discussion of results aimed at evaluating the architecture's operational efficiency.

Chapter 4: Optimal Intelligent Edge-Servers Placement in the Healthcare Field In this chapter, the second contribution is presented. With the intent of enhancing the HMAN architecture, two improvements are proposed through the implementation of two algorithms. This is followed by the presentation of outcomes aimed at assessing algorithms' validation.

Chapter 5: An Adaptive SDN-Based Load Balancing Method for Edge/Fog-Based Real-Time Healthcare Systems In this chapter, the third contribution is presented. This contribution introduces an innovative method for load balancing. Subsequently, the chapter presents results aimed to evaluate and discuss the performance of the two algorithms.

Chapter 6: Conclusion and Future work

In this final chapter, the study is summarised, and the research questions are answered based on the findings and analysis. The conclusions drawn from the research are highlighted, emphasizing their implications and contributions to the field. Additionally, future research directions and recommendations for further investigations are provided, identifying areas that require further exploration or refinement.

Bibliography

This section provides a comprehensive list of all the sources cited in the thesis, following a specific citation style.

Chapter 2

Literature Review

2.1 Introduction

The Internet of Things (IoT) stands as a groundbreaking concept that establishes seamless connections between unique physical and virtual devices, employing diverse communication protocols. According to statistical data, the number of IoT devices linked through wireless technology is predicted to soar to 50 billion by the year 2025 [5]. These devices encompass a wide range of possibilities, including smartphones, bio-nano devices, body sensors, smart tags, wearable gadgets, embedded objects, and traditional electronic devices. Many of these devices are equipped with an array of sensors responsible for gathering vital environmental data, forming the bedrock of data-driven intelligence. As a result of this extensive proliferation of IoT devices, there is a tremendous surge in data generation. In many cases, this data exhibits the following characteristics: (i) it is only locally relevant; (ii) it requires additional analysis and processing; (iii) the result of processing and subsequent actuation is subject to strict latency requirements; and (iv) the raw data is ephemeral, meaning it is no longer relevant after processing and can thus be discarded or moved to persistent storage. Examples of such areas include real-time video analytics, cognitive aid applications, mobile gaming, and autonomous driving. To provide valuable insights to users, the collected data requires comprehensive processing and analysis. However, due to inherent limitations, lightweight IoT devices face challenges in handling complex computations.

One promising solution is cloud computing, where IoT data is transmitted to a cloud server for processing, and the subsequent results are sent back to the devices. The success and rapid adoption of cloud computing have opened up numerous business opportunities. Industry projections suggest that the global cloud computing market is set to achieve a momentous milestone, reaching a value of £1 trillion by the year 2024 [6]. However, relying solely on cloud computing to transmit collected data and await processing responses entails certain disadvantages.

One notable concern is the cost of data transmission, which places additional strain on network bandwidth and resources. As data volume increases, performance may deteriorate, rendering this approach less suitable for time-sensitive applications. Domains such as smart transportation, electricity grid management, smart cities, and healthcare applications, which require ultra-short response times, may find cloud computing unsuitable. The geographic distance between users and most data centres contributes to undesirable delays. Moreover, the limited availability of resources and bandwidth exacerbates the issue, leading to significant network latency, an undesirable characteristic for time-sensitive applications. Consequently, relying solely on cloud computing presents a critical challenge [7].

Given these concerns, it becomes crucial to explore complementary approaches to mitigate the limitations of cloud computing. Investigating strategies that aim to enhance data transmission efficiency, reduce network latency, and improve responsiveness for time-sensitive applications is of utmost importance. The adoption of alternative paradigms, such as edge computing or fog computing, presents viable solutions to address the challenges associated with exclusive reliance on cloud-based architectures in time-sensitive contexts. Edge computing embodies an innovative computing paradigm that redirects computational data, applications, and services away from central cloud servers towards the network edge. Embracing edge computing allows content providers and application developers to deliver services to users in closer geographical proximity, leading to accelerated response speeds. In short, edge computing circumvents the expensive transfer of data that requires low latency and high bandwidth to the cloud by storing and processing the data on diverse, nearby resources [8, 9].

2.2 Edge Computing Paradigm

Edge computing is a distributed computing paradigm that brings computation and data processing capabilities closer to the source of data, enabling real-time processing and low-latency responses. Unlike traditional cloud computing, where data is sent to remote data centres for analysis, edge computing occurs at or near the edge of the network, right where the data is generated by devices or sensors [10].

By moving computation closer to the edge devices, edge computing addresses the challenges posed by the increasing volume of data generated by Internet of Things (IoT) devices and the need for rapid, real-time decision-making. This proximity to data sources reduces the dependency on constant high-speed internet connectivity and minimises latency, which is crucial for applications that require instantaneous responses and smooth user experiences. Edge computing is particularly relevant in various industries, such as industrial automation, autonomous vehicles, healthcare, and retail. In these scenarios, critical decisions must be made locally and promptly to ensure safety, efficiency, and overall system reliability [11].

In an edge computing architecture, each edge device can perform data processing and analytics independently or collaborate with nearby edge nodes to share resources and insights. Additionally, with advancements in edge hardware and software technologies, devices can perform complex tasks that were traditionally reserved for centralised cloud servers [12].

While edge computing provides numerous advantages it also poses challenges related to security, scalability, and managing a decentralised infrastructure. Nevertheless, as the Internet of Things continues to grow and demand for real-time processing increases, edge computing's role as a fundamental component of the modern computing landscape becomes more prominent [13].

It is noteworthy that edge computing is not an alternative solution to cloud computing; rather, it complements and extends the capabilities of cloud computing. Both edge computing and cloud computing serve different purposes and cater to distinct use cases. While cloud computing excels in providing centralised data storage, vast computing resources, and extensive data analysis, edge computing focuses on bringing computation closer to the edge devices, enabling real-time processing, and reducing latency. Together, they form a powerful and versatile computing ecosystem, addressing a wide range of applications and ensuring efficient, responsive, and secure data processing across various scenarios [14]. Later, we will further explore these technologies through a comprehensive comparison, including cloud computing and other prominent solutions.

2.2.1 Edge Computing Architecture

Edge architectures refer to the various design and deployment models used to implement edge computing solutions. Edge architectures provide flexible solutions for a variety of applications, from simple single-node setups to distributed and hierarchical models that can handle large-scale and diverse edge computing requirements. Each architecture offers specific advantages based on the use case, network size, and desired performance characteristics. Edge architectures can vary in complexity and scalabil-

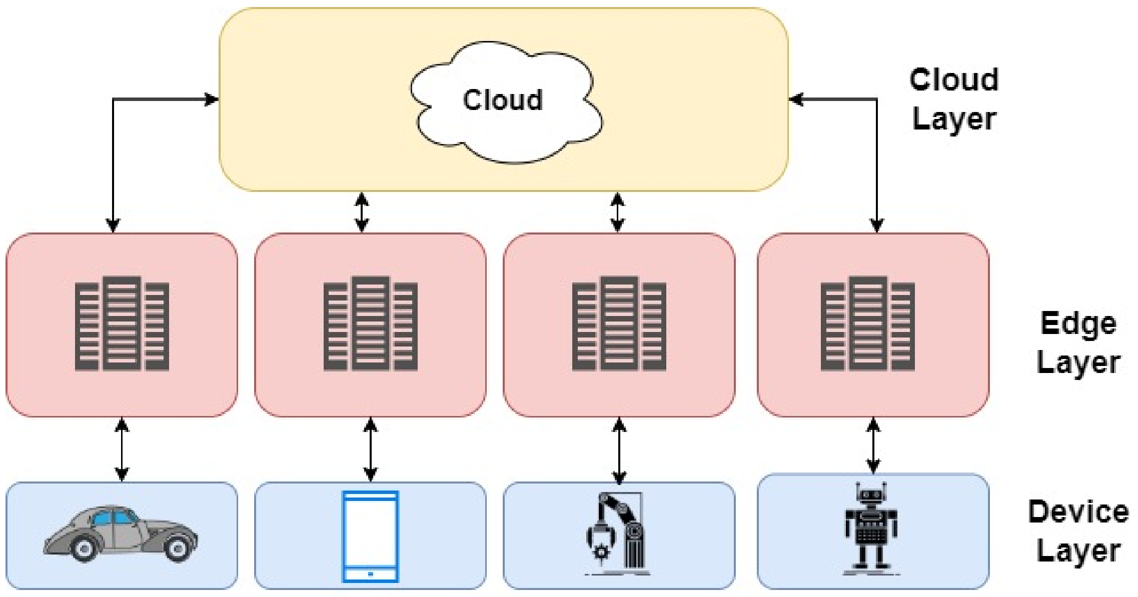


Figure 2.1: Single-Node Edge Architecture

ity, depending on the specific use case and the size of the network. An in-depth explanation of some common edge architectures will be provided in the following [15, 16, 17, 18, 19, 20].

2.2.1.1 Single-Node Edge Architecture

This is the simplest form of edge architecture, where a single edge device or server performs all the data processing and computation at the edge. It is suitable for smaller-scale deployments or isolated edge computing applications. The single-node architecture is cost-effective and straightforward to implement, making it ideal for scenarios where limited resources and low-latency processing are sufficient. Figure 2.1 depicts a diagram illustrating a Single-Node Edge architecture.

2.2.1.2 Distributed Edge Architecture

In this architecture, multiple edge devices or servers are deployed across the network, distributed geographically to serve various edge computing requirements. These nodes collaborate to process data and share computational resources. Distributed edge ar-

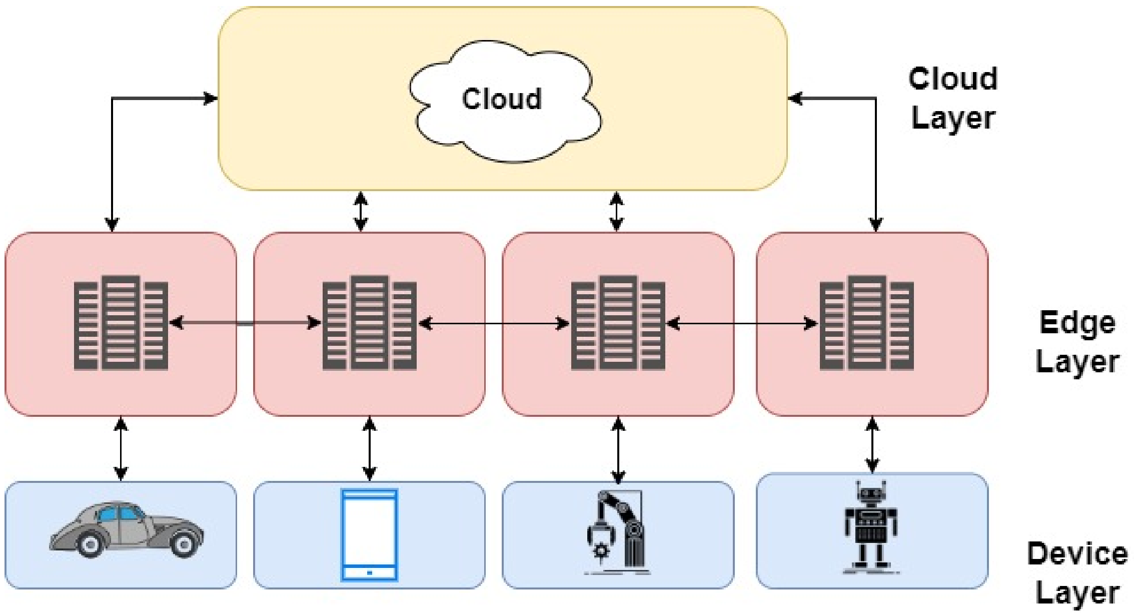


Figure 2.2: Distributed Edge Architecture

chitectures offer greater scalability, fault tolerance, and load balancing. Data can be processed locally at the nearest edge node, alleviating the load on the central infrastructure. Figure 2.2 illustrates a diagram depicting a Distributed Edge architecture.

2.2.1.3 Hierarchical Edge Architecture

Hierarchical edge architectures combine elements of both centralized cloud computing and distributed edge computing. It involves intermediate layers of edge nodes placed between the edge devices and the cloud. In this model, the intermediate edge nodes aggregate and preprocess data from multiple edge devices before sending relevant information to the central cloud. This reduces the amount of data transmitted to the cloud, minimising latency and bandwidth requirements. Figure 2.3 illustrates a diagram depicting a Hierarchical Edge architecture.

2.2.2 Access Technologies and Communication Patterns

Edge computing, as a distributed computing paradigm, relies on various access technologies and communication patterns to establish seamless connectivity between edge

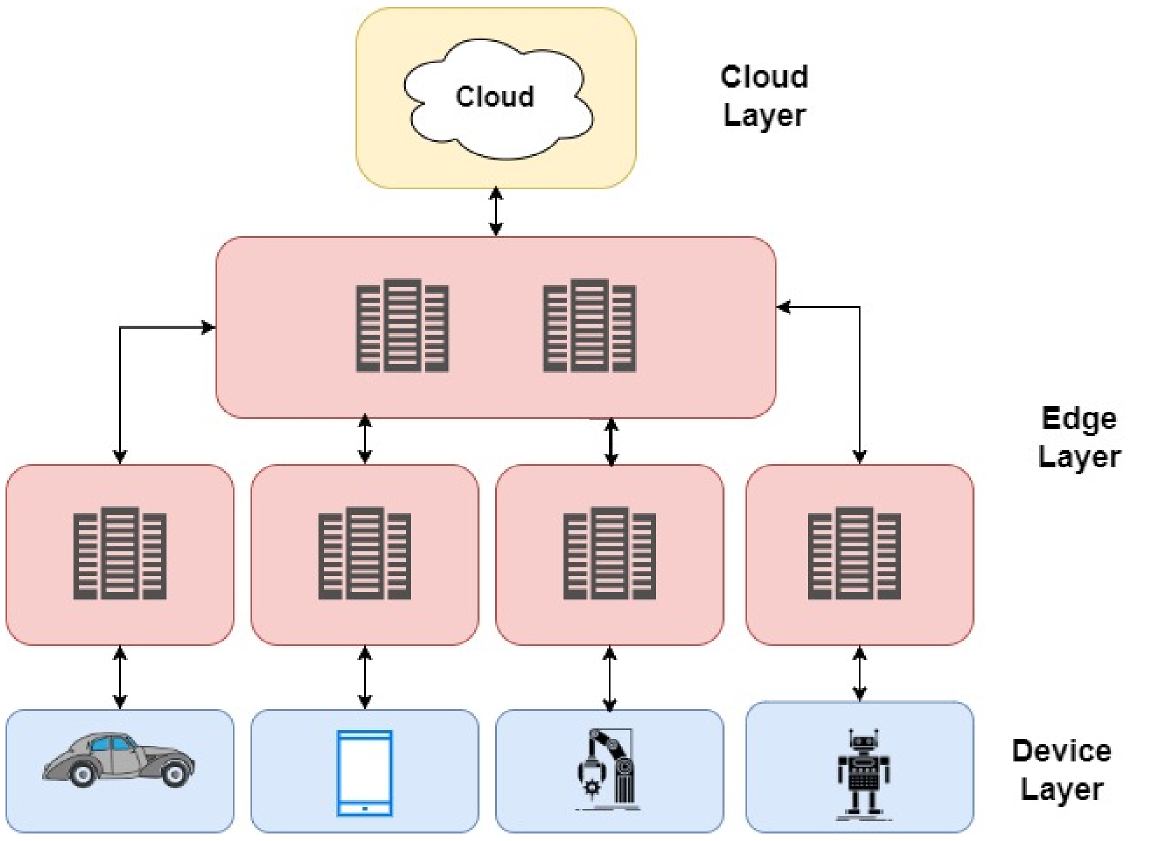


Figure 2.3: Heirarchical Edge Architecture

devices, edge nodes, and the central cloud. These technologies play a pivotal role in enabling real-time data processing, reducing latency, and ensuring efficient communication across the network.

2.2.2.1 Access Technologies

1. Wireless Technologies: Wireless access technologies, such as 5G, Wi-Fi 6, and Bluetooth Low Energy (BLE), are fundamental to edge computing deployments. 5G, with its low-latency and high-bandwidth capabilities, is particularly well-suited for time-sensitive applications, while Wi-Fi 6 offers enhanced data transfer rates and improved efficiency in crowded environments. BLE is commonly used for low-power, short-range communication in IoT devices, making it ideal for edge computing scenarios.

2. Wired Technologies: In certain edge computing environments, wired access technologies like Ethernet and Fiber Optics are employed to ensure reliable and high-speed data transmission. These technologies offer stable connections and higher bandwidth, making them suitable for applications that require constant data streaming and minimal latency [21, 22].

2.2.2.2 Communication Patterns

1. Device-to-Edge Communication: Edge computing involves direct communication between edge devices and the edge nodes. This pattern enables local data processing and decision-making, minimising the need for data transmission to the central cloud. Device-to-edge communication is essential for applications requiring real-time responses and low-latency processing, such as industrial automation and autonomous vehicles.

2. Edge-to-Edge Communication: In certain edge architectures, edge nodes collaborate and communicate with each other, sharing resources and data insights. Edge-to-edge communication allows for load balancing, fault tolerance, and localized data aggregation, resulting in a more efficient and resilient edge computing infrastructure.

3. Edge-to-Cloud Communication: Edge computing is designed to complement cloud computing, and therefore, edge nodes communicate with the central cloud for more extensive data analysis, long-term storage, and resource-intensive computations. Edge-to-cloud communication ensures that valuable data can be leveraged by the cloud to provide deeper insights and support applications that require comprehensive analytics [23, 24].

2.2.3 Device Ecosystem

The device ecosystem within the realm of edge computing plays a pivotal role in enabling the seamless integration of a diverse array of interconnected devices. With mobile broadband subscribers surpassing 6 billion in 2019 and expected to exceed 8 billion by 2024, the proliferation of edge devices has surged, driven by the transformative potential of the Internet of Things (IoT) in creating interconnected smart surroundings.

The IoT's objective is to interconnect various devices, enabling them to communicate and collaboratively analyse data to generate smart insights. Among the numerous edge devices, smartwatches, smart eyewear, personal on-body sensors, and a host of other end devices contribute to a deluge of data that demands further processing, storage, and sharing to deliver new services. The device ecosystem is marked by remarkable heterogeneity, not only in terms of functions and form factors but also in capabilities and computational capacity. End devices alone exhibit diversity, producing substantial volumes of data that necessitate advanced processing. The range of devices capable of performing edge computations extends from consumer-grade hardware to powerful data center-grade hardware. Notably, small-scale single-board computers, such as Raspberry Pis, have found applications in edge computing, alongside home routers and compact setups equipped with more robust hardware.

This device heterogeneity poses one of the most significant challenges in edge computing. The collaborative use of diverse devices in proximity to the mobile client allows for distributed edge computing. Every device capable of spare computational resources becomes a candidate for edge computing, emphasising the versatility and adaptability of the device ecosystem in facilitating real-time data processing. To provide an overview of the diverse edge computing device landscape, Figure 2.4 depicts an insightful visual representation. The intricacy of the device ecosystem, encompassing a myriad of interconnected edge devices, end devices, and computation-performing devices, underscores the robustness of edge computing's architecture in addressing a wide range of applications and computing requirements. The device ecosystem in edge computing represents a dynamic and diverse landscape, fuelled by the exponential growth of mobile broadband subscribers and the proliferation of IoT-enabled devices. The collaborative synergy between different devices allows for edge computing's seamless integration, empowering the creation of smart surroundings and delivering cutting-edge services. Embracing device heterogeneity as an opportunity, edge computing stands poised to revolutionize computing paradigms and drive innovations across numerous industries [25].

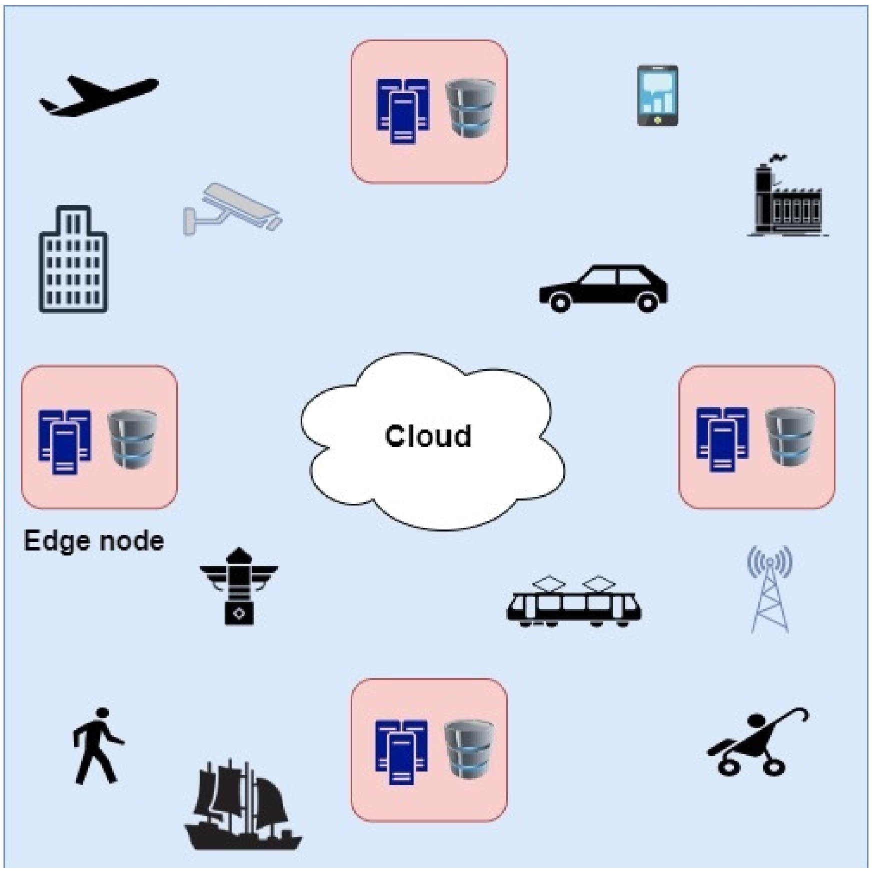


Figure 2.4: Edge Ecosystem

2.2.4 Edge Computing Implementation

To implement the architecture of edge computing outlined in the preceding sections, various research efforts have focused on the design of edge computing models. Among these models, two dominant approaches have emerged: the Hierarchical Model and the Software-Defined Model.

2.2.4.1 Hierarchical Model

The Hierarchical Model organises the edge architecture into a hierarchy, considering the deployment of edge/cloudlet servers at different distances from the end users. This hierarchical structure defines functions based on proximity and available resources. By dividing the edge computing network into tiers, the hierarchical model effectively describes the network structure of edge computing [26, 27]. The advantage of this approach lies in its ability to manage resources efficiently, optimise data processing based on proximity, and ensure seamless communication between edge nodes and end devices.

2.2.4.2 Software-Defined Model

Given the scale of edge computing, with numerous applications and millions of end users and devices, managing edge computing for the Internet of Things (IoT) can be exceptionally complex. To address this challenge, the Software-Defined Model comes into play. Software-Defined Networking (SDN) offers a viable solution to cope with the intricacies of edge computing management [28, 29]. By abstracting the underlying network infrastructure, SDN enables centralised control and programmable management of edge devices and edge nodes. This approach streamlines the management process, enhances scalability, and facilitates dynamic resource allocation, making it suitable for the large-scale and dynamic nature of edge computing environments.

2.3 Edge computing and similar concepts

Edge computing has gained significant prominence in recent years as a transformative paradigm in the field of computing. It has emerged as a compelling solution to address the challenges posed by centralized cloud computing and the proliferation of Internet of Things (IoT) devices. Several related concepts and computing paradigms have surfaced, each offering unique approaches to decentralized data processing, low-latency applications, and improved user experiences. In this section, we explore edge computing and its relationships with other similar concepts [30, 31, 32, 33, 34, 35, 36, 37].

2.3.1 Cloud Computing

Cloud computing serves as a foundation for edge computing, and the two concepts are closely interconnected. While cloud computing offers vast storage, computing resources, and extensive data analysis capabilities, it inherently faces challenges concerning data latency and bandwidth constraints. Edge computing complements cloud computing by bringing computation closer to the edge devices, alleviating the strain on the central cloud and enabling real-time data processing. The combination of edge computing and cloud computing forms a powerful and versatile computing ecosystem, catering to diverse use cases across industries. Figure 2.5 illustrates the prevalent architecture of cloud computing.

To gain further insight, Table 2.1 provides a comparative analysis between edge computing and cloud computing.

2.3.2 Fog Computing

Fog computing is another related concept that shares similarities with edge computing. The term "fog computing" was introduced to describe an intermediate layer

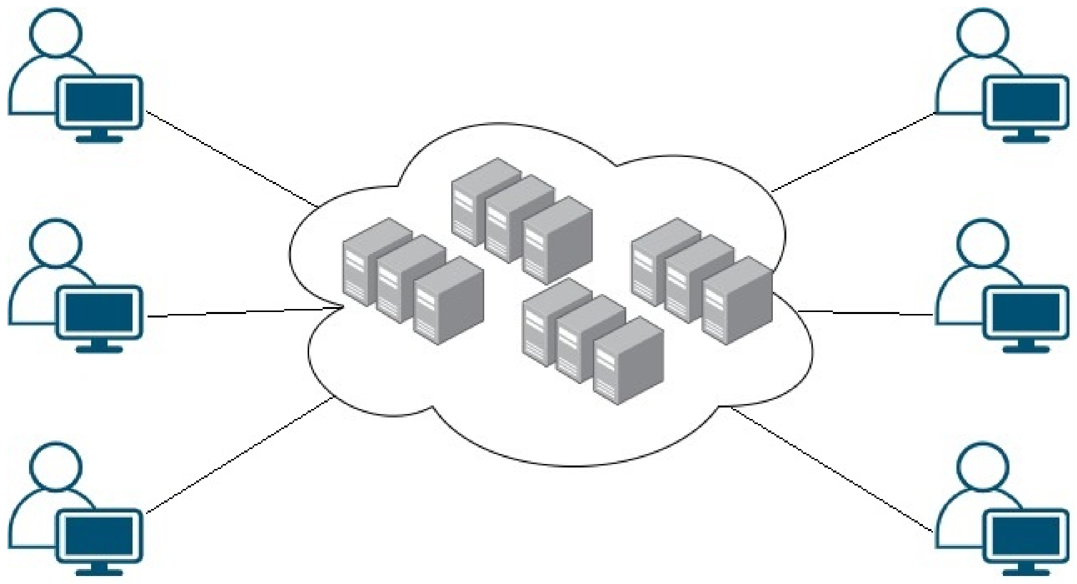


Figure 2.5: Cloud Computing Architecture.

Table 2.1: A comparison between edge and cloud computing

Criteria	Cloud computing	Edge Computing
Proximity to users	Low	High
Latency	High	Low
Infrastructure	centralised	Distributed
Hardware Heterogeneity	Low	High
Number of Locations	Few	Many
Resources at individual locations	Many	Few
Availability and reliability	High	Low
Connection to resources	Long-thin	Short-fat
Applications	Data-driven	User-driven

between edge devices and the central cloud. Fog nodes are deployed at the edge of the network and provide computing, storage, and networking capabilities to support edge devices. The key distinction lies in the level of abstraction; edge computing emphasises localised data processing at the device level, while fog computing focuses on an intermediate layer that offers additional services closer to the end-users. Despite this difference, both paradigms seek to minimise latency, enhance user experiences, and improve the overall efficiency of data processing. Figure 2.6 illustrates the prevalent architecture of fog computing.

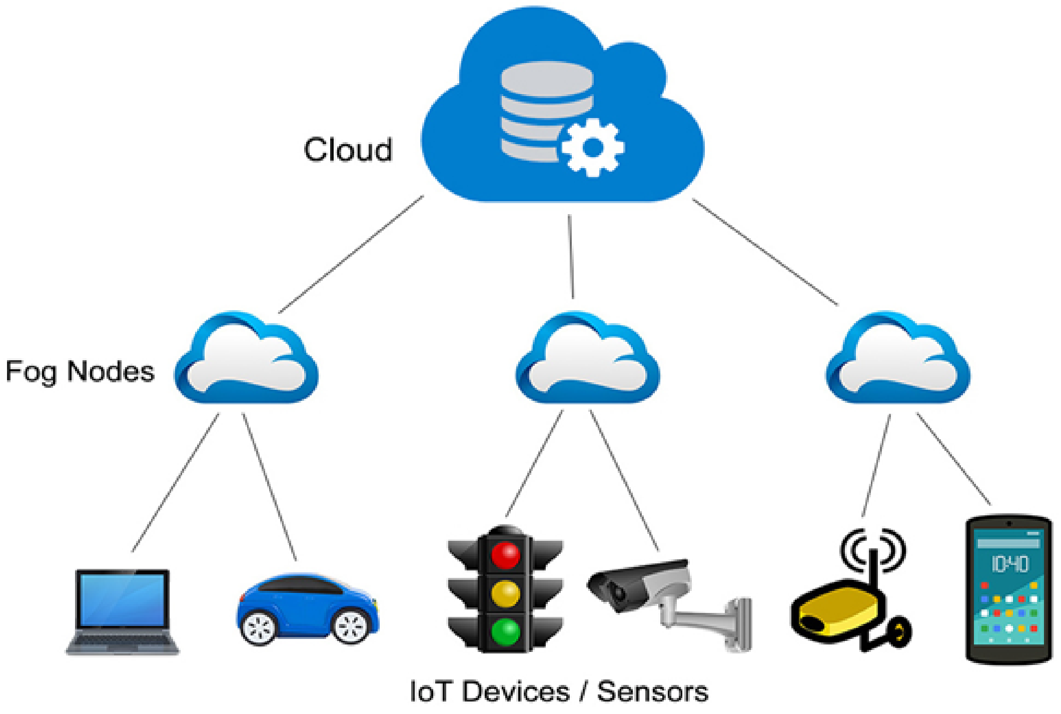


Figure 2.6: Fog Computing Architecture.

2.3.3 Mobile Edge Computing (MEC)

Mobile Edge Computing (MEC) is a subset of edge computing that concentrates on providing computing capabilities and services at the edge of the mobile network. MEC servers are deployed at base stations, enabling mobile users to access low-latency, edge-based services. MEC enhances the performance of mobile applications, such as augmented reality and real-time video streaming, by reducing latency and alleviating the burden on the central cloud. While edge computing has a broader scope, encompassing various types of edge devices, MEC specifically caters to mobile users and applications, ensuring efficient data processing for time-sensitive mobile services. Figure 2.7 illustrates the prevalent architecture of MEC computing.

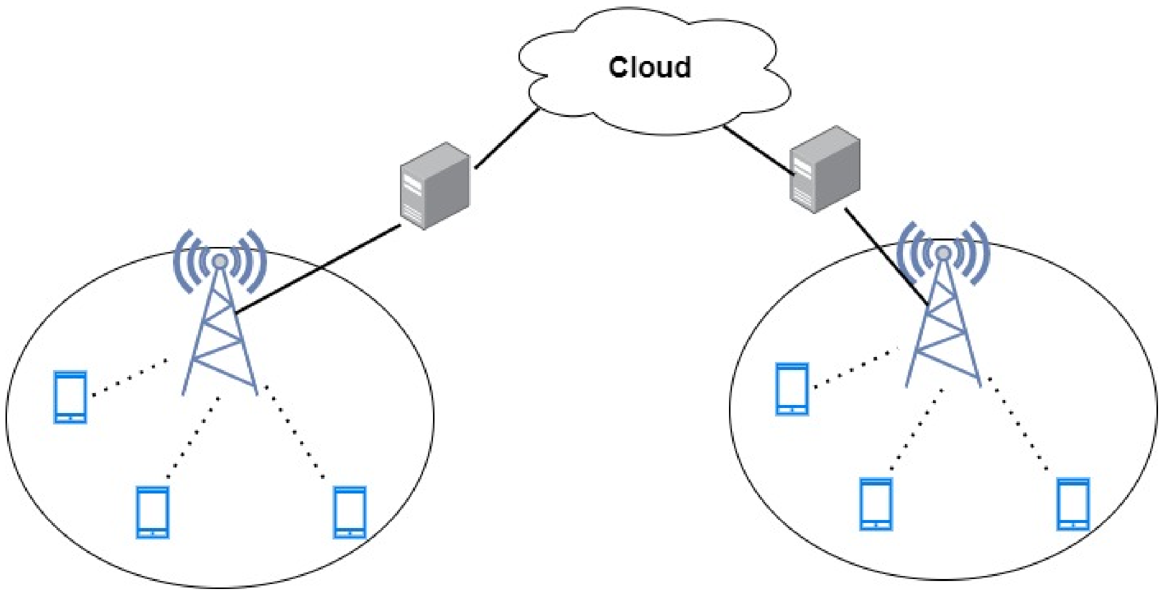


Figure 2.7: MEC Architecture.

2.3.4 Cloudlet

Cloudlet is another concept closely related to edge computing, offering a specific approach to address latency and data processing challenges. Cloudlet, as a mobile micro data center situated at the edge of the network, enriches the edge computing landscape by providing localised computing resources and reducing latency for time-sensitive applications. Its collaboration with edge devices and compatibility with edge computing make it a valuable component in the distributed computing ecosystem. By optimising data processing and enhancing application performance, cloudlet plays a crucial role in shaping the future of edge computing, driving innovations and creating more efficient and responsive services. When integrated with other similar concepts, such as fog computing, mobile edge computing, and decentralized computing, cloudlet contributes to the advancement of decentralised computing paradigms, paving the way for a connected and intelligent future. Figure 8 illustrates the prevalent architecture of cloudlet computing.

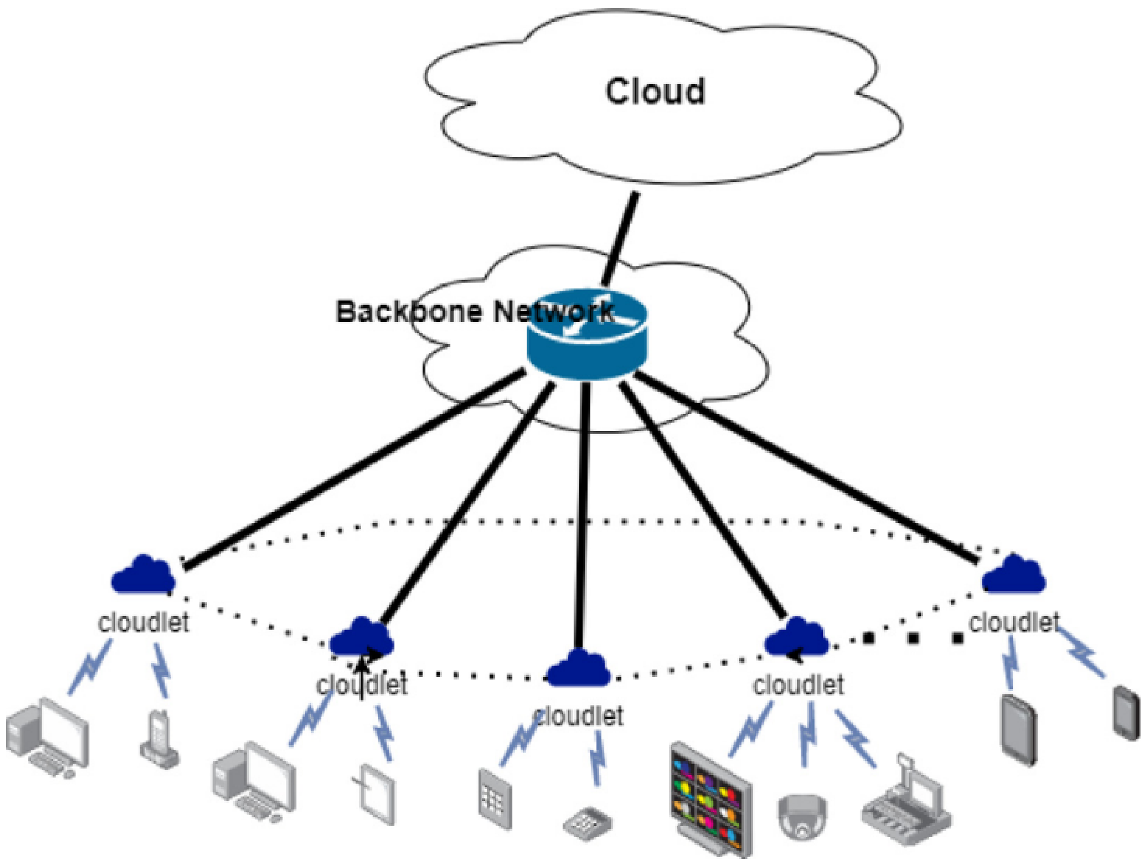


Figure 2.8: Cloudlet Architecture.

2.3.5 Decentralised Computing

Decentralised computing is a broader term that encompasses various paradigms, including edge computing, fog computing, and other distributed computing models. It emphasises the distribution of computing tasks and data processing across multiple nodes, moving away from traditional centralised architectures. Edge computing aligns with the decentralised computing philosophy by pushing computation to the network's periphery, closer to the data source. This distribution of computing resources contributes to improved scalability, reduced single points of failure, and enhanced responsiveness.

2.4 Enabling Technologies and techniques

Edge computing has emerged as a revolutionary computing paradigm, driven by a diverse set of enabling technologies and techniques that enhance its capabilities. In this section, we explore some of the key enabling technologies and techniques in the context of our study, focusing on their relevance and importance in optimising edge computing deployments.

2.4.1 Offloading Techniques

Offloading is a fundamental technique in edge computing, which involves the transfer of computation tasks from resource-constrained edge devices to more powerful edge servers or cloud resources. By offloading computation, edge devices can conserve energy and extend their battery life, while also benefiting from the higher processing capabilities of edge servers. This technique is particularly crucial for latency-sensitive applications, enabling real-time data processing and enhancing the overall efficiency of the edge ecosystem [38, 39, 40, 41].

2.4.2 Artificial Intelligence (AI) Integration

AI plays a pivotal role in edge computing by enabling intelligent decision-making at the edge of the network. By integrating AI algorithms into edge devices and edge servers, edge computing environments can perform local data analysis, inference, and predictions. AI-driven edge computing enables autonomous systems, real-time analytics, and personalised services. It also reduces the dependency on centralised cloud resources, making edge devices more self-sufficient and efficient [42, 43, 44].

2.4.3 Optimal Servers Placement

Optimal servers placement involves strategically locating edge servers within the network to achieve optimal performance and resource utilisation. By carefully selecting server locations, data can be processed closer to end-users, reducing latency and improving response times. Optimal placement considers factors such as the density of edge devices, data traffic patterns, and geographical distribution to ensure efficient data processing and minimise network congestion [45, 46, 47].

2.4.4 Software-Defined Networking (SDN)

SDN is a networking technology that centralises the control of network resources through software-based controllers. In the context of edge computing, SDN allows for dynamic network reconfiguration and resource allocation to accommodate changing edge computing demands. SDN enhances network flexibility, scalability, and adaptability, enabling efficient communication between edge devices, servers, and cloud resources [48, 49, 50].

2.4.5 Load Balancing

Load balancing is a critical technique in edge computing to evenly distribute computing tasks and data processing across multiple edge servers. By balancing the workload, load balancing ensures that no single server is overwhelmed with requests, preventing performance bottlenecks and ensuring optimal resource utilisation. Load balancing techniques can be adaptive and responsive to real-time changes in data processing demands, optimising the edge computing environment [51, 52, 53].

2.5 Advantages of Edge Computing

Edge computing offers a multitude of advantages that have positioned it as a transformative computing paradigm in various industries. As data generation continues to soar, and the demand for real-time and low-latency applications grows, edge computing has become a crucial enabler for the efficient processing and delivery of data. In this section, we explore the key advantages of edge computing in more detail, highlighting its impact on the technological landscape and its wide-ranging benefits [54, 55, 56, 57].

2.5.1 Reduced Latency and Improved Responsiveness

Edge computing significantly reduces latency by processing data closer to the source, at the edge of the network. This proximity ensures faster data processing and delivery, leading to improved application responsiveness. Latency-sensitive applications, such as online gaming, live video streaming, healthcare, and autonomous vehicles, greatly benefit from edge computing's ability to deliver real-time results.

2.5.2 Bandwidth Optimisation and Network Efficiency

Edge computing optimises bandwidth usage by performing data processing and filtering locally. Only relevant data is transmitted to centralised cloud servers, reducing the amount of data traversing the network. This efficient data handling minimises network congestion, conserves bandwidth, and enhances overall network efficiency.

2.5.3 Enhanced Data Privacy and Security

Edge computing enhances data privacy and security by keeping sensitive data closer to the source. Since data is processed locally or within the organisation's premises, there is reduced exposure to potential cyber threats during data transmission to external cloud servers. This proximity also allows for better control over data access, ensuring compliance with data privacy regulations.

2.5.4 Resilience and Reliability

Edge computing improves system resilience and reliability by enabling applications to function even during network disruptions or connectivity issues with the central cloud. Local edge servers can store and process critical data, ensuring the availability of essential services in edge computing environments. This resilience is particularly beneficial for applications in remote or harsh environments.

2.5.5 Scalability and Flexibility

Edge computing offers scalability and flexibility, allowing organisations to easily expand their computing resources by adding edge servers as the demand grows. This dynamic scalability ensures that edge computing environments can adapt to changing workloads and application requirements. Edge computing's flexibility also allows for the deployment of specialised edge devices to meet specific application needs.

2.5.6 Offline Operation and Edge Intelligence

Edge computing enables certain applications to operate offline, even when disconnected from the central cloud. By processing data locally, edge devices can continue functioning and providing essential services in scenarios with limited or no internet connectivity. Edge intelligence allows edge devices to analyse and respond to data locally, reducing dependency on cloud connectivity.

2.5.7 Real-time Data Analytics and Insights

Edge computing enables real-time data analytics and insights, allowing organisations to extract valuable information from data as it is generated. By analysing data at the edge, businesses can make faster and data-driven decisions, leading to improved operational efficiency and competitive advantages.

2.5.8 Lower Operating Costs

Edge computing can lead to cost savings by reducing data transmission costs to centralised cloud servers. Offloading computation from edge devices to edge servers also extends the battery life of edge devices, reducing maintenance and replacement costs.

2.6 Challenges of Edge Computing

While edge computing offers numerous advantages, it also faces several challenges that need to be addressed for its widespread adoption and successful implementation. As the technology continues to evolve, it is essential to acknowledge and mitigate these challenges to fully realise the potential of edge computing. In this section, we explore the key challenges faced by edge computing in more detail, highlighting their impact on the technology landscape and the strategies to overcome them [58, 59, 60, 61].

2.6.1 Latency Variability

Edge computing relies on proximity to end-users to reduce latency. However, the variability of network conditions and edge server loads can lead to unpredictable latency. Maintaining consistent low-latency performance across different edge nodes requires effective load balancing and resource management. Advanced algorithms and predictive analytics can be employed to dynamically allocate computation tasks and ensure optimal response times.

2.6.2 Limited Computing Resources

Edge devices, such as IoT devices and sensors, often have limited computing capabilities and storage capacities. This limitation can hinder the execution of resource-intensive applications at the edge. Efficient task offloading, lightweight computation, and AI-based optimisations are essential to make the most of these constrained resources. Additionally, the adoption of edge servers with higher processing power and memory can help overcome resource limitations.

2.6.3 Data Management and Security

Processing data at the edge raises concerns about data management and security. Ensuring data integrity, confidentiality, and compliance with privacy regulations becomes crucial. Robust encryption, secure communication protocols, and access controls are necessary to protect sensitive data at the edge. Implementing secure data lifecycle management practices and data anonymisation techniques can further enhance data security.

2.6.4 Interoperability

As edge computing environments grow, ensuring interoperability between various edge nodes and cloud resources becomes challenging. Standardisation of protocols and frameworks, along with seamless integration of heterogeneous devices, can facilitate smooth scaling and efficient communication. Open standards like MQTT and CoAP can foster interoperability in IoT-based edge computing systems.

2.6.5 Edge-to-Cloud Orchestration

Coordinating computation and data flow between edge devices and central cloud servers requires efficient orchestration. Balancing the processing load between edge and cloud resources while considering network conditions is essential to achieve optimal performance. The adoption of edge computing management platforms and edge-to-cloud orchestration tools can streamline this coordination process.

2.6.6 Edge Device Management and Maintenance

Managing a large number of edge devices spread across different locations poses operational challenges. Remote device management, software updates, and diagnosing device failures demand efficient device monitoring and maintenance mechanisms. Edge device management platforms that enable remote monitoring, automated updates, and predictive maintenance can ease these challenges.

2.6.7 Data Aggregation and Consistency

Aggregating data from various edge nodes while maintaining data consistency can be complex, especially in dynamic edge environments. Ensuring synchronised data across edge servers and the central cloud is crucial for accurate data analysis and decision-making. Consistency protocols like CRDTs (Conflict-free Replicated Data Types) can be employed to maintain data integrity across distributed edge nodes.

2.6.8 Edge-to-Cloud Security Handoff

Secure data transmission between edge devices and the central cloud is vital. However, the handoff of data between the edge and cloud introduces potential security vulnerabilities. Secure handoff mechanisms, such as secure tunnels and authentication protocols, are needed to mitigate risks. Additionally, secure communication channels and encryption keys can enhance data security during transmission.

2.7 Edge Computing Applications

Edge computing has unlocked a diverse range of applications across various industries, reshaping data processing, service delivery, and user experiences. The proximity of computation and data processing to the network's edge facilitates real-time insights, minimises latency, and enhances the handling of extensive data volumes. In this section, we delve into some of the key edge computing applications that have revolutionised industries and empowered innovative solutions [62, 63, 64, 65, 66, 67, 68].

2.7.1 Industrial Internet of Things (IIoT)

Edge computing is at the forefront of the Industrial Internet of Things (IIoT) revolution. By deploying edge devices within manufacturing plants and industrial settings, real-time data from sensors and machines can be processed locally, enabling predictive maintenance, fault detection, and process optimisation. IIoT applications powered by edge computing enhance production efficiency, reduce downtime, and ensure seamless operations in critical industrial processes.

2.7.2 Autonomous Vehicles

Edge computing plays a pivotal role in the development of autonomous vehicles. Processing sensor data and making critical decisions in real-time is crucial for the safety and functionality of self-driving cars. Edge computing enables onboard computation, reducing the reliance on cloud connectivity for immediate responses to changing road conditions and potential hazards.

2.7.3 Smart Cities

Edge computing is a cornerstone of smart city initiatives. By deploying edge servers and devices throughout the city infrastructure, real-time data from sensors, cameras, and IoT devices can be processed locally. This enables intelligent traffic management, waste management optimisation, environmental monitoring, and improved public safety.

2.7.4 Agriculture and Precision Farming

Edge computing is empowering precision farming by enabling real-time monitoring of crop conditions, soil moisture, and weather data. Edge devices in smart farming applications optimise irrigation schedules, automate pest control, and enable precision agriculture practices, leading to increased crop yields and resource efficiency.

2.7.5 Augmented Reality (AR) and Virtual Reality (VR)

Edge computing is instrumental in delivering seamless AR and VR experiences. By processing intensive AR/VR applications at the edge, latency is minimised, resulting in smooth and immersive user experiences. This capability is critical for AR/VR applications in gaming, training, and remote collaboration.

2.7.6 Healthcare Application

In the healthcare sector, edge computing has revolutionised telemedicine and patient monitoring by enabling real-time operations due to its inherent advantage of ultra-low latency. Medical devices deployed at the edge can efficiently collect and process vital signs and patient data locally, empowering healthcare professionals to monitor patients remotely in real-time. This instantaneous feedback facilitates timely medical interventions and significantly reduces the burden on centralised healthcare systems.

Edge computing rapidly analyses real-time bio-signals from sensor nodes at edge data centers, providing healthcare professionals with valuable insights for proactive decision-making. Its capabilities ensure efficient patient monitoring and timely interventions, improving healthcare services and outcomes. As technology evolves, further integration is expected to advance medical practices, making healthcare more efficient and patient-centric.

Beyond real-time monitoring, edge computing brings forth advanced techniques and services to the healthcare industry. Embedded data mining, distributed storage, and alerting services are among the key capabilities made possible through edge computing integration. These services enhance data processing efficiency and promote seamless communication of critical information for timely decision-making. The next section presents a compilation of related work that pertains directly to our study's focal areas in the upcoming chapters. This compilation encompasses investigations into healthcare architecture, offloading methodologies, optimal server placement strategies, and load balancing techniques. These insights from existing research will serve as a foundation for the subsequent exploration and analysis of our study.

2.8 Related Work

2.8.1 Healthcare Architecture and Offloading Techniques

In order to review the publications from the most reliable sources, a systematic literature review (SLR) technique is used in this research. Finding, interpreting, and evaluating research findings that address the research questions is the major goal of the SLR. Both automated and manual searches were performed to gather the research findings from primary studies. The primary studies underwent a quality assessment in order to analyse the data and find the most appropriate results. The first step in the SLR is to define and document the search strategy. Inclusion and exclusion criteria of the research papers are the next steps. Then, quality criterion assessment is defined, while quantitative meta-analysis is the last step in the process.

Based on the existing literature, the idea of using wearable sensors to monitor patients has been investigated for many years. It started with the use of PC-based stations, right up to the use of modern technological solutions such as the IoT technologies that have proven their ability to effectively develop healthcare at various levels. A large number of research works examine developments in the key enabling technologies for the IoMT, which include edge/fog computing, wearable medical devices, communication networks, and cloud computing. For example, Asif-Ur-Rahman et al. [69] proposed a heterogeneous IoHT framework consisting of five layers. Although mobile healthcare data can be offloaded to the cloud for processing, analysis and storage, offloading data to remote clouds still results in excessive latency. Furthermore, privacy is not addressed when offloading, placing sensitive health data at risk of external attacks. The number of possible scenarios for data transmission and processing paths is limited, which raises concerns about system scalability and service availability.

In the same way, M. Ahmad et al. [70] proposed a framework of Health Fog where the cloud and patients are separated by a layer of fog computing to reduce the E2E extra communication cost. The framework was provided by a cloud access security broker (CASB) to enhance data privacy and security. In [71], the researchers presented a privacy preserving healthcare system for data management in cloud. The blockchain technology was used to store all medical data to increase privacy. T. Muhammed et al. [72] proposed a four-layer ubiquitous healthcare framework based on edge computing technology to optimise data rate, data caching and data decisions. A cloud-fog-based IoT healthcare framework was structured in reference [73] to optimise the latency issues when cloud computing was used only to process the offloaded healthcare data. However, sending medical personal data outside the network increases privacy concerns and latency issues with the increase in the data size.

Another system for patient monitoring is suggested by A. M. Rahmani et al. [74], which introduced a fog-computing-based healthcare system architecture integrated with smart e-Health gateways. The strategic position of these gateways at the edge of the network was exploited to present a Smart e-Health Gateway through offering some important services, such as local storage, real-time local data processing and data mining.

A different solution by C. Kai et al. [75] that investigated the collaborative computation offloading, computation and communication resource allocation schemes and developed a collaborative computing framework to improve the cloud-edge-end task processing efficiency whilst maintaining limited computation and communication capabilities. A pipeline-based offloading strategy was proposed to partially process the collected data at the terminals, edge nodes and cloud. Reference [76] presented BEdgeHealth, a decentralised architecture that combined MEC with blockchain for data offloading and sharing amongst distributed nodes. Rajasekaran et al. [77] suggested an autonomous monitoring system model to provide healthcare services. This model uses the analytical hierarchy process for equitable distribution of energy amongst the nodes. The results demonstrated that the proposed model could support a large number of nodes with less energy.

There are also other applications in patients monitoring: I. Azimi et al. [78] introduced a portable detection and prediction monitoring system of the patients' health deteriorations that can be used at Hs or at homes. G. Muhammad et al. [79] presented a pathology monitoring system by using a cloud-based IoT and a machine learning classifier. Scalability, secure transmission and availability are amongst the concerns that still need to be addressed. S. He et al. [80] proposed an IoT-enabled medical services framework called FogCapCare to detect patented heart health conditions by integrating a cloud layer with a sensor layer. Reference [81] presented a Fog-enabled Technique for Clinical Healthcare framework based on edge computing, deep learning and automated monitoring to deliver highly useful real-life healthcare systems, such as cardiology.

In summary, the research studies reviewed here provided system architectures to improve data collection, management, and processing. These systems have reflected good performance in providing health services. However, concerns regarding the lack of data flow and offloading scenarios to increase the possibility of local data processing continue to persist. This situation mainly leads to other concerns, such as latency, data privacy, service availability and scalability issues. Therefore, the main objective of this research is to design a ubiquitous and scalable healthcare architecture with a high level of data management flexibility and system availability, while increasing privacy and capacity, and reducing latency. An effective solution is to develop the system to be able to handle different workloads with various data processing scenarios by adopting a cooperative hierarchical structure. The architecture presented in this research is unparalleled in combining the concept of hierarchical Edge/Fog computing with unique cooperative offloading techniques (HOSSC algorithm) between the architecture units exploiting many data flow scenarios to reflect additional options of data processing paths within the network. This combination can create a scalable and ubiquitous health system applicable at cities by utilising (for the first time) the already existing infrastructure (e.g., MCs and Hs). The proposed architecture takes advantage of the MCs' and Hs' geographical locations to provide electronic health services whilst increasing the computational capacity and the quality of healthcare with greater privacy and reduced latency. Table 1 presents a comparison of the proposed architecture with the previous existing ones.

Table 2.2: Feature comparison table of HMAN (proposed) with the previous existing architectures

References	**Structure**		**Features**				
	Hierarchical	**Cooperative**	**Scalability**	**Availability**	**Capacity**	**Privacy**	**Latency**
Ref. [69]	No	No	N/A	N/A	Yes	No	Yes
Ref. [70]	No	No	N/A	N/A	N/A	Yes	Yes
Ref. [71]	No	No	Yes	N/A	N/A	Yes	No
Ref. [72]	No	No	N/A	N/A	Yes	Yes	Yes
Ref. [73]	No	Yes	N/A	N/A	Yes	No	Yes
Ref. [74]	No	No	N/A	Yes	No	No	No
Ref. [75]	No	No	No	No	No	Yes	No
Ref. [76]	No	Yes	No	N/A	No	Yes	Yes
Ref. [77]	Yes	No	N/A	N/A	Yes	No	Yes
Ref. [78]	No	No	No	No	Yes	No	Yes
Ref. [79]	Yes	No	No	No	No	No	No
Ref. [80]	No	No	N/A	N/A	No	No	Yes
Ref. [81]	No	Yes	N/A	N/A	No	Yes	Yes
Proposed HMAN	Yes	Yes	Yes	Yes	Yes	Yes	Yes

2.8.2 Optimal intelligent Edge-Servers Placement Study

As this study can be divided into two parts, we have divided the related work section into two subsections: priority mechanism and edge-servers placement approaches.

2.8.2.1 Priority Mechanism Approach

In recent years, significant advancements in IoT and body sensor devices, along with the integration of emerging technologies such as edge or fog computing with the cloud, have spurred numerous research projects dedicated to developing IoT-based smart health monitoring frameworks. However, existing healthcare systems, exemplified by studies [82, 83, 84, 85, 86, 87, 88, 89, 90] and the baseline study [HMAN] which published in [91], often adopt a uniform approach to processing and predicting data, following a first-in-first-out concept, without considering the urgency of cases. As a result, patients with severe illnesses may encounter delays in receiving timely treatment. To tackle this challenge, this study proposes enhancements to the HMAN healthcare system by incorporating an AI-based data classification method. The main objective is to establish a priority mechanism based on the urgency of patients, with the aim of improving both the Quality of Service (QoS) and the Quality of Experience (QoE) in healthcare delivery. By introducing a prioritisation framework that considers the urgency of cases, this approach seeks to ensure that critical patients receive prompt attention and care.

2.8.2.2 Edge-Servers Placement Approach

The placement of edge-servers, commonly referred to as cloudlets, is a crucial factor that significantly impacts the efficiency of a system. However, this topic has received limited attention in the existing literature, with only a few studies addressing it. The key question revolves around determining the optimal placement of edge-servers in a system to maximize benefits while considering different goals, such as reducing costs or minimising latency.

Several papers have explored server placement in large-scale environments, specifically the wireless metropolitan area network (WMAN), and are particularly relevant to our study. Jia et al. [92] investigated how to place a number of cloudlets and allocate users among them in a way that minimises the average system response time. To address this issue, they also suggested the density-based clustering (DBC) approach. To reduce the average cloudlet access delay, Zhao et al. [93] suggested using a ranking-based heuristic for K cloudlets deployment. In order to reduce the latency between the users and cloudlets, Xu et al. [94] proposed an exact solution to the problem by placing K cloudlets in strategic locations within a large-scale WMAN.

Similarly, a cost-aware cloudlet deployment technique was suggested by Fan et al. [95] to improve the trade-off between deployment cost and latency. To minimise communication latency, Meng et al. [96] proposed some algorithms for deploying cloudlets in a group of access points and routing mobile task requests. Firstly, they derived an approximation algorithm to choose candidate locations based on historical data. Then, they presented an iterative algorithm to select the best locations to deploy cloudlets in. Finally, an online job routing algorithm was proposed to route the request to the cloudlet with minimum latency. Yao et al. [97] proposed a low-complexity heuristic algorithm to cost-effectively deploy cloudlets without compromising a pre-determined QoS. They essentially assumed that the cloudlet servers are heterogeneous, meaning that they have different cost and resource capacities. A novel framework named EdgeON was presented by Santoyo-Gonzalez et al. [98], which was aimed at reducing the total cost when placing and operating the edge-servers network. To reduce the average response time, Li et al. [99] suggested two methods for edge-servers placement: flat and hierarchical. They found that the hierarchical approach has a better performance in reducing system response time. A different solution by Li et al. [100] presented the problem of an energy-aware edge-servers problem as a multi-objective optimisation one. Then, they suggested a particle swarm-based energy-aware algorithm for reducing energy consumption in computing resource utilisation.

There are also other studies in edge-servers placement. The edge-server placement problem was formulated, firstly, by Lahderanta et al. [101] as a constrained optimisation mode to reduce the sum of weighted distances between the edge-servers and access points. Then, they designed the PACK algorithm to minimise the distances while balancing the load among servers. Lovén et al. [102] proposed a new algorithm to choose the optimal number of edge-servers to be placed and optimally re-allocate access points, accordingly, in order to improve QoS.

In addition to the aforementioned studies, recent papers have made significant contributions to the field of edge-server placement. For instance, [103] proposed novel approaches for addressing the challenges in deploying edge servers. Their work focused on optimizing the placement of edge servers in order to minimize latency and enhance network performance. Through innovative algorithms and techniques, they achieved improved efficiency and effectiveness in edge-server deployment, paving the way for further advancements in this area. In [104], cloudlet deployment in IoT networks was investigated with the aim of optimizing deployment cost and network latency. The authors proposed a fault-tolerant cloudlet deployment scheme based on software-defined network technology. Their binary-based differential evolution cuckoo search algorithm showed promising performance in terms of cost and latency optimization. Lastly [105] focused on deploying edge servers effectively and economically in wireless metropolitan area networks. They addressed the problem of minimizing the number of edge servers while ensuring specific QoS requirements by extending the definition of dominating set and formulating it as a graph theory problem. Their proposed greedy-based algorithms showed feasibility and effectiveness in achieving efficient edge-server deployment. These papers contribute to the advancement of edge-server placement and offer valuable insights for future research in this area.

To summarise, the reviewed research studies present innovative methods for placing edge-servers, showcasing notable achievements in terms of cost and/or time. The literature has explored four main perspectives regarding edge-server placement:

1. Minimising response time (latency) by employing various distance measures.
2. Minimising the cost of server deployment while maintaining a maximum acceptable delay.
3. Optimising the trade-off between delay and costs.
4. Maximising coverage, i.e., the number of users served.

In comparison, this study introduces a novel (unprecedented) approach that involves placing a flexible number of servers to minimise costs while ensuring acceptable latencies based on patient conditions through a priority mechanism. It presents a fresh perspective on selecting and locating edge-servers within a specific area, considering different variables such as the number of connections between competing sites, distances, and historical workloads. Furthermore, this study stands out in the literature as the first to address the problem of edge-server placement specifically in the healthcare domain.

2.8.3 Load Balancing Techniques

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. [106] 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. [107] 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 minimising the average response time of the Traffic Management Server (TMS) highlights the importance of efficient offloading and real-time management in IoV environments. Ning et al. [108] 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. [109]. Their Dynamic Load Balancing (DLB) approach, utilising the Ull Multiobjective Framework (UllMF), aims to optimise 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. [110] addressed the load balancing problem by proposing a strategy for task allocation based on intermediary nodes. Their multi-step process, involving classification, evaluation, and task assignment, contributes to improved load balancing and resource utilisation.

Considering the resource optimisation aspect, Nayyer et al. [111] developed the LBRO (Load Balancing for Resource Optimisation) approach. It tackles both resource scarcity and under-provisioning issues in cloudlets, maximising resource utilisation at the cloudlet level. Their approach ensures stable resource utilisation without compromising application performance, demonstrating the potential for enhancing resource efficiency in edge computing systems. Similarly, He et al. [112] introduced an enhanced constrained particle swarm optimisation algorithm within the 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. [113] 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 optimising resource allocation and performance in edge computing environments.

Furthermore, Shahrbabaki et al. [114] proposed an SDN-enabled scheme for load balancing in edge nodes used for real-time IoT video data analytics. By utilising 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 re-routing 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. [115] 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 optimise task scheduling and adjust operations. The method achieves load balancing through centralised and decentralised decision-making based on global and local information, respectively. By incorporating these decision-making models, TDBEC provides an efficient load balancing mechanism for multi-edge collaboration in WMANs.

In summary, the existing methods in the literature can be mainly categorised into static, dynamic, and occasionally centralised or decentralised load balancing techniques. Previous research in this field has predominantly focused on addressing the load balancing problem through separate utilisation of these methods. However, this approach has demonstrated limitations in terms of adaptability, real-time responsiveness, and resource efficiency.

In contrast to prior studies, the proposed algorithm integrates these diverse approaches, combining the simplicity of static methods with the adaptability and responsiveness of dynamic methods. Furthermore, it incorporates localised decision-making, centralisation, and network collaboration or decentralisation across different areas. The proposed approach incorporates a load balancing algorithm for initial load distribution decisions and an SDN-based algorithm to offload data within the network, either locally within the areas or between them. This integrated 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.

2.9 Summary

This chapter offers a thorough examination of Edge Computing, encompassing its architectural principles, enabling technologies, advantages, challenges, applications, and a special focus on its integration within the healthcare domain.

The **Edge Computing Paradigm** section explores into crucial concepts related to edge computing technology, including its architecture, access technologies, ecosystem, and implementation.

In the **Edge Computing and Similar Concepts** section, we distinguish Edge Computing from cloud computing, highlighting their complementary roles and distinct use cases. Additionally, we provide a comparative evaluation of edge computing with related paradigms like fog computing, mobile edge computing (MEC), and cloudlet computing, emphasizing their unique attributes for specific applications.

Enabling Technologies and Techniques are discussed in another section, highlighting on pivotal technologies for Edge Computing, such as offloading techniques, artificial intelligence (AI) integration, optimal server placement, Software-Defined Networking (SDN), and load balancing. These technologies empower Edge Computing applications, making them highly suitable for various industries, including healthcare.

The section on **Advantages and Challenges of Edge Computing** showcases its real-time insights, reduced latency, improved privacy, and enhanced system efficiency. We also address challenges, such as latency variability, resource limitations, and data management and security.

Edge Computing Applications section demonstrates its transformative impact across diverse sectors, ranging from smart cities to industrial automation. Notably, we explore its role in healthcare applications, where integrating Edge/Fog computing yields a multitude of advantages, ultimately improving patient outcomes.

In the **Related Work** section, we explore into relevant literature, shedding light on subjects like healthcare architecture, offloading methods, optimal server placement, and load balancing techniques. These insights serve as a foundation for our study's subsequent exploration and analysis.

Chapter 3

Towards a cooperative hierarchical healthcare architecture using the HOSSC algorithm

3.1 Introduction

The Internet of Things (IoT) phrase was coined in 1999 by Kevin Ashton, a British technology pioneer [116]. Nowadays, the phrase 'Internet of Things' refers to any network of devices or things that can be characterized as a system connecting objects, involving humans, animals, and inanimate objects, over the Internet. The number of IoT applications have been exponentially growing [117, 118]. According to Gartner's research, the worldwide number of connected devices could increase from 15.1 billion in 2020 to 29 billion in 2030 [119]. The McKinsey Global Institute estimated that IoT applications will have an annual global economic impact ranging from \$3.9 trillion to \$11.1 trillion [120]. According to Facts and Factors, the IoT market was expected to reach \$1842 billion by 2028, rising at a Compound Annual Growth Rate of 24.5 percent from 2021 to 2028 [121].

The Internet of medical things (IoMT) is one of the most attractive areas in the recent IoT developments, with a promising business growth forecast of \$158.07 billion by 2028 [122]. In certain ways, the IoMT might be considered one of the future's economic foundations. Furthermore, a wide range of services can be provided through a variety of IoMT applications. Hence, massive data collected from endless IoMT sensors or devices must be processed at powerful data centres to provide additional insight to users and service providers [123].

In the case of near-real-time applications, the widespread adoption of cloud-based infrastructure may exacerbate the real-time constraints and burden the network infrastructure from the on-premises gateway to the cloud. Meanwhile, simple networking approaches are rarely feasible when attempting to provide healthcare services to a wide large of patients due to the complex nature of healthcare. For instance, patient privacy prevents data from being stored in a public cloud. Another issue to address is the patient's safety as data must be instantly available with a predetermined delay and intolerant cloud failures.

The emergence of cloud computing technology is creating new business opportunities, with the global market for this technology expected to exceed $1 trillion by 2024 [124]. However, relying on cloud computing to transfer data and wait for a response after the data are processed has several drawbacks. With regard to the data transmission cost, this process places additional burden on the network in terms of the required bandwidth and resources, resulting in degraded performance as the data volume increases.

This situation is considerably worse for time-sensitive applications, such as smart transportation, electricity grid, smart city and some healthcare applications, wherein a short response time is not negotiable. Cloud computing cannot meet these standards because most data centres are located far away from users, causing delays. In addition to limiting available resources and bandwidth, cloud computing implementation might result in significant unacceptable network latency for time-sensitive applications [125].

To address these challenges, Edge/Fog computing paradigms have been introduced as complementary technologies for cloud computing. The primary goal of Edge/Fog computing is to shorten the distance between users and data processing centres (servers). Accordingly, various advantages are realized, such as faster response time. In addition, placing the servers at the same user network can improve the bandwidth and privacy level because the data can be stored and processed locally. This feature highlights how useful this technology would be for most applications, especially time-sensitive ones, such as healthcare [126].

According to projections, the world's population will reach 9.7 billion people in 2050 [2]. Overcrowding in cities poses serious threats to the quality of life to those who live there. According to WHO, poor healthcare management contributes to 2.6 million fatalities per year, and the rate is rising [3, 4]. Consequently, cities will face several challenges in the near future if appropriate solutions are not implemented. One of the major problems is the potential stress on the health system in cities. To address such issues, the emerging technology that allows humans and devices to work and communicate using IoT, Edge, Fog and Cloud technologies is the electronic healthcare (e-healthcare) system. This system can be a reasonable alternative to the traditional healthcare techniques based on the rapid advancements in communication and information technologies.

This work introduces a unique cooperative hierarchical architecture for the healthcare system in cities based on Edge/Fog computing technology supported by the HOSSC algorithm that provides adaptable offloading techniques. The presented architecture, named as Healthcare Metropolitan Area Network (HMAN), connects the patients in a city to the healthcare system by utilising infrastructures that already exist in that city, such as medical centres (MCs) and hospitals (Hs). Consequently, this will provide an easy way to decide where to place servers since these facilities are spread all over the cities. This work mainly contributes to a ubiquitous and scalable e-healthcare system capable of delivering health services to patients without restrictions of space and time. The HMAN system manages the data within the network as much as possible through the HOSSC algorithm and various offloading scenarios, resulting in less delay and high privacy levels. The system networks patients, physicians and family members to cooperatively monitor the patients and remotely obtain regular updates on their health conditions. In addition, relying on the hierarchical architecture is promising to gain an accumulative computational capacity because the data are cooperatively processed at multiple system units before reaching the cloud. Furthermore, the HMAN system also contributes to increased availability because the designed scenarios ensure multiple methods to process the data, hence avoiding singular unit failures.

Chapter Organisation: The remainder of this chapter is organised as follows: Section 3.2 introduces the proposed HMAN architecture; Section 3.3 presents the data flow algorithm and the offloading scenarios in the suggested architecture; Sections 3.4 and 3.5 describe the proposed HMAN system and the offloading model, respectively; Section 3.6 presents and discusses the simulation results; Section 3.7 identifies system limitations and possible improvements in the future; Finally, Section 3.8 provides a summary of the chapter.

3.2 Proposed HMAN architecture

One of the biggest challenges the healthcare providers can face is to meet the needs of people as their multiple health conditions worsen. These people require extensive support from healthcare providers because they may have a lower quality of life than others and a higher risk of premature death than usual [127]. However, a lack of patient status information hampers the ability of healthcare providers to meet these needs [127]. Moreover, any latency in providing the necessary medical response makes the physicians unable to deliver all the preventive services recommended by the US Preventive Services Task Force. According to [128], preventive services require roughly 37 min/year per child and 40 min/year per adult. Accordingly, providing high-quality healthcare services to these targeted patients at MCs or Hs requires more time than what these caregivers can afford [129]. On this basis, designing an advanced ubiquitous e-healthcare system has become crucial to fulfilling these aspirations through local monitoring and data processing. Based on the primary analysis of the locally collected data, the outcome of such an advanced system can help healthcare providers in managing their resources in providing services for the people in need.

This section describes the HMAN, a cooperative hierarchical architecture for the healthcare system based on Edge/Fog computing technology supported by a collaborative offloading algorithm (HOSSC). The HMAN system aims to provide a mobile 24 h monitoring service for patients in need. Furthermore, the suggested framework is promising to meet the healthcare important requirements, such as availability, scalability, communication delay, computational capacity, and privacy. The main idea of the HMAN architecture design is based on the procedure of the existing standard healthcare systems. The integration of the existing infrastructure in the city (e.g., MCs and Hs) with the recent technology (i.e., Edge/Fog computing) can play a key role in achieving a robust system and providing its services to all patients in any city.

For example, most health systems, certainly in the USA and the UK, have two main levels of health services delivery, namely, primary and advanced. The primary level is provided by MCs distributed around cities to provide primary care for patients. Individuals that need to be examined or treated with a higher degree of expertise are filtered by these primary care facilities. Meanwhile, most Hs tend to provide advanced, more specialised, and skilled care to patients referred to them by the MCs. Accordingly, the proposed architecture can be built on these two layers and additional layers, as demonstrated in Figure 3.1. Based on the figure illustration, the HMAN architecture consists of four layers, an IoT layer, an Edge/Fog layer, a cloud layer and an application layer. A further detailed explanation of these four layers is provided in the following parts of this section.

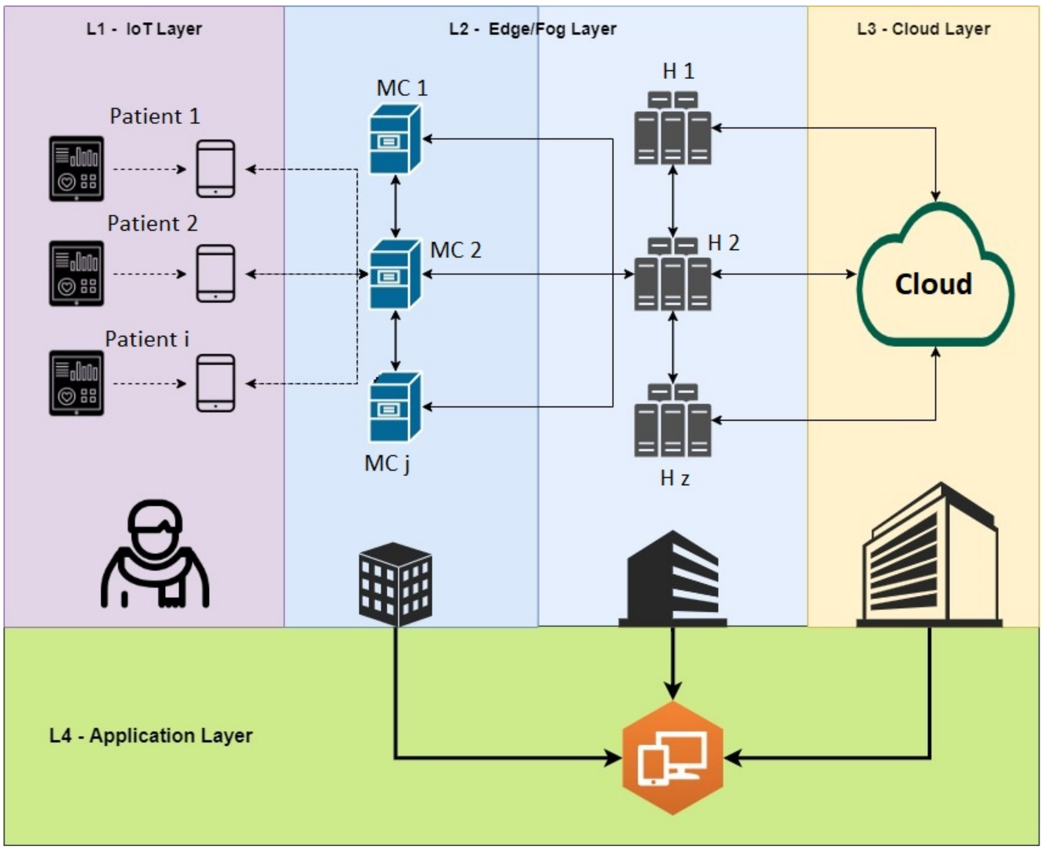


Figure 3.1: The HMAN Architecture. The arrow directions indicate the flow of data. MC = Medical centres, H = Hospitals. The MCs and Hs are hierarchically connected

3.2.1 IoT Layer

The IoT layer is the first layer of the HMAN architecture. The incorporation of WBANs and innovative technologies in the various layers of IoT ecosystems has many benefits, such as enhancing storage and data availability while lowering data transmission latency [7]. Therefore, this layer consists of vital signs monitoring sensors, including on body or/and in body (implants) sensors, and a smartphone (or any wearable devices). The sensors and Wireless Body Area Network are connected to the patient's body to sense vital signs and send the sensed data to the smartphone through a wireless link (e.g., Bluetooth or Zigbee). The smartphone is responsible for receiving the data from the sensors and performing a basic preprocessing and initial data analysis (i.e., aggregation, fusion, filtering, and classification). Consequently, the received data are classified into normal and abnormal according to a predefined threshold assigned based on the patient's conditions. A physician may assign a threshold value for a diabetic patient, for instance, based on the patient's history record to take further actions. Meanwhile, the normally classified data are temporarily locally stored with no actions required, whilst the abnormal data can be offloaded to the next layer for further actions. Moreover, the offloaded data generate an alert to the patient and a family member to notify the condition.

3.2.2 Edge/Fog Layer

The Edge/Fog layer is the next layer in the architecture, where the local MCs and the Hs are hierarchically connected to cooperatively accommodate the patients' needs by the allocation of the nearest resources. This layer of the hierarchy's servers, which are often geo-distributed desktops or workstations, receives workloads directly from patients' smartphones through wireless connections. The HMAN suggests that each MC is connected to the nearest two MCs, and each group of MCs is linked to a local H through optical fibre links. Meanwhile, each H is connected to two neighbouring Hs through optical fibre links, whilst each H is linked to the cloud via the Internet backbone. This cooperative hierarchy architecture summarises the main contribution of this study on how the data offloading can be collaboratively managed between the neighbouring MCs and Hs and how it can help the system gain more computational capacity. If the workloads received by an MC exceed its computational capacity, then the overload amount of data is further offloaded either to a neighbouring MC or to the local H. The same method is applied to the local H when dealing with incoming data. Consequently, the forwarded data inherit more computational capacity gained at the servers of these facilities based on the hierarchy architecture. More detailed scenarios of the data flow are provided in Section 3.3 to emphasise the cooperative hierarchy of data processing along with the accumulatively gained computational capacity. Assigning only two neighbouring facilities to back up the local MCs and Hs achieves the aimed inherited computational capacity while sharing the data within as fewer local parties as possible and maintain a simplistic, applicable and cost-effective system. Concurrently, this neighbouring units' collaboration reduces the distance and latency.

3.2.3 Cloud Layer

The cloud layer is the third layer in the architecture, which represents the furthest layer the data can reach according to the data flow scenarios detailed in Section 3.3. Specifically, it is the worst case of the proposed offloading techniques happening only when the workload overloads all the previous layers' capacity. Hence, the overload data must be sent to the cloud.

3.2.4 Application Layer

The application layer is the topmost layer of the HMAN architecture. The user interface between the relevant members and the system itself is provided by this layer. Moreover, the system administrator can access the system resources through this layer.

3.3 HOSSC - Data flow and offloading algorithm

Computational offloading is a distributed paradigm for transferring all or a portion of the data from local servers to remote ones to speed up data processing and conserve energy. However, this process has several conditions. First, local execution cannot be done due to the limited resources of the local servers. Second, the offloading time, including the communication time and the remote execution time, is less than the local execution time, as expressed in equation (3.1) [130].

$$T_{offloading} < T_{local} \tag{3.1}$$

Algorithm 1 presents an explanation of the proposed HOSSC algorithm at different layers of the suggested system. The algorithm determines whether or not the data offloading is necessary between the HMAN units after considering the time latency in equation 3.1 based on the size of the workload received by an MC or an H compared with the maximum data processing capacity of these facilities. To better understand the HOSSC algorithm, as illustrated in Figure 3.2, the following step-by-step data flow and offloading algorithm from the data collection to the final outcome is described in the next part of this section.

The monitored patient's data are collected with a smartphone and classified into normal and abnormal based on a pre-set threshold assigned by healthcare specialists. The normal data are temporarily stored for further future analysis of the patient's records. Meanwhile, the abnormal data are immediately sent to the local MC, and an alert to a family member is initiated. Once the abnormal data reaches the nearest MC in the patients' area, that MC processes the data directly if the available resources are capable to do so. However, if the received workload requires greater computational resources than the MC's capacity, then the MC processes the data within its capacity and offloads the overload data to the connected neighbouring MCs. The only possibility that data must be forwarded to the local H is when both neighbouring MCs are busy.

The same cooperative processing in between MCs is applied to the Hs to manage the received data by the local H. The received data is processed within the H server's capacity unless it requires more computational capacity. If so, the overload data can be then forwarded to the neighbouring cooperative Hs. Therefore, the data are mostly managed within the MCs and the Hs without reaching the cloud. The data arrives the cloud only if all local and neighbouring MCs and Hs cannot accommodate the overload data. On this basis, the possible abnormal data flow and offloading scenarios are follows:

- **Scenario 1**: The abnormal data collected by IoT devices require less processing capabilities than what the local MC can afford. Hence, the local MC manages to process the data.
- **Scenario 2**: The data require more processing capabilities than that available at the MC. Accordingly, the data are partly processed at the MC, whilst the rest are offloaded to the nearest idle neighbouring MC. This scenario assumes that the offloaded overload data can be accommodated by the processing capacity of at least one neighbouring MC. Otherwise, scenario 3 is applied.
- **Scenario 3**: The data require more processing capabilities than that available at the MC and at the neighbouring centres. Accordingly, the data are partly processed at the local MC, whilst the rest are offloaded to the local H. This scenario assumes that the offloaded overload data can be accommodated by the processing capacity of the local MC and the local H. Otherwise, scenario 4 is applied.
- **Scenario 4**: The data require more processing capabilities than that available at the local MC and the local H. Accordingly, the data are partly processed

at the local MC and H, whilst the rest are offloaded to the nearest idle neighbouring Hs. This scenario assumes that the offloaded overload data can be accommodated by the processing capacity of at least one H of the neighbouring ones. Otherwise, scenario 5 is applied.

- **Scenario 5**: The data require more processing capabilities than that available at the cooperative MCs and Hs. According, the data are partly processed at the local MC and H, whilst the rest are offloaded to the cloud. This scenario expresses the only possibility that the data can reach the cloud. Figure 3.3 visualises the five scenarios.

Having addressed the possible data flow and offloading scenarios, the HMAN framework can contribute to enhancing the provided services by:

- Offering a high level of patients' privacy by processing the data locally without reaching the cloud as much as possible. This step is met through cooperative data processing and offloading in between facilities.
- Increasing the computational capacity through the hierarchical connection of the facilities in the Edge/Fog layer.
- Decreasing the system response time (i.e. the latency) as a result of the greater possibilities of local data processing shortening the distance between the patients and the healthcare providers.
- Maintaining high availability of service because the presented system does not rely on one scenario to forward and process the data. Meanwhile, the HMAN sustains a reliable service by providing multiple scenarios that can cooperatively process the data within the system's units to ensure that the service is constantly maintained for patients.

Algorithm 1: HMAN offloading scenarios and SRT calculation algorithm

```
Input: $Ps, r_{Fib}, \lambda_i, T_{Wi}, T_{AP}, B, \mu_{MC}, \mu_H, \mu_{Cloud}, \lambda_{MC}, \lambda_H$
Output: Present five offloading scenarios and facilitate the SRT calculation.
Assumptions: Each MC is connected to two neighbouring MCs,
and each H is connected to two neighbouring Hs.

1- Data (workload) at the local MC.
a- Calculate: $\lambda_{Sum} \leftarrow \sum_i^{Ps} \lambda_i$        The local MC's workload
b- Calculate: the data fraction scalar K1
if $\lambda_{Sum} <= \lambda_{MC}$ then:
    K1 $\leftarrow$ 1      (means all $\lambda_{Sum}$ will be processed at the local MC and nothing
                           will be offloaded.)(Scenarios 1).
else then:
    $K1 \leftarrow \frac{\lambda_{MC}}{\lambda_{Sum}}$      (means a part of data ($K1 * \lambda_{Sum}$) will be offloaded to the
                           higher units.)
end if statement.
In both cases, calculate: the MC's queuing time
$fQ((K1 * \lambda_{Sum})$ & the processing time.
Then, calculate the response time:
$t_{MC} \leftarrow fQ + processingtime$

2- When $\lambda_{Sum} > \lambda_{MC}$
a- Calculate: $\lambda_j \leftarrow (1 - K1) * \lambda_{Sum}$
$\lambda_j$ = The local MC workload which should be offloaded to:
1- one or both Idle neighbours,
or 2- the local H
b- Calculate the optic fiber latency:
$V \ and \ S \leftarrow (\lambda_j * 8)/r_{Fib} + 4.9 \ microseconds * Distance(kilometer)$
Now, there are two cases:
1- $\lambda_j <= \lambda_{MC}$      (means it is possible to send data to one of the neighbouring
```

MCs **(Scenario 2)**. Therefore, the local MC will first check for an Idle neighbouring MC.

2- $\lambda_j > \lambda_{MC}$ means the data must be offloaded to the local H.

Case 1: $\lambda_j <= \lambda_{MC}$

Calculate the response time in MC t_{Mc} or Hs t_H based on the state of neighbouring MCs:

1- $[0, 0] \rightarrow$ Both neighbours are busy, then the MC offloads the data to the H **(Scenario 3)**.

2- $[1, 0] \rightarrow$ neighbour 1 is idle, then the MC offloads the data to the MC (neighbour 1) **(Scenario 2)**.

3- $[0, 1] \rightarrow$ neighbour 2 is idle, then the MC offloads the data to the MC (neighbour 2) **(Scenario 2)**.

4- $[1, 1] \rightarrow$ Both neighbours are idle, then the MC offloads the data to the nearest one **(Scenario 2)**.

Case 2: $\lambda_j > \lambda_{MC}$

That means the local MC must offload the λ_j data to the local H.

Now, again there are two cases:

Case A: $\lambda_j <= \lambda_H$ (All λ_j data is processed at the local H.) **(Scenario 3)**.

$K2 \leftarrow 1$

Case B: $\lambda_j > \lambda_H$ (A part of the λ_j data should be offloaded to the higher units.)

$K2 \leftarrow \frac{\lambda_H}{\lambda_j}$

In both cases, calculate: the queuing time $(fQ((K2 * \lambda_j))$ & processing time in the local H.

Then, calculate the response time

$$t_H \leftarrow fQ + processingtime.$$

3- When $\lambda_j > \lambda_H$

a- **Calculate:** $\lambda_z \leftarrow (1 - K2) * \lambda_j$

λ_z = The local H's workload which should be offloaded out of the local H to:
1- one or both Idle neighbours, or 2- the cloud.

b- **Calculate the fiber optic latency**

$$L \leftarrow (\lambda_z * 8)/r_{Fib} + 4.9\, microseconds * Distance(kilometer)$$

Now, λ_z must be offloaded according to two cases:

Case 1: $\lambda_z <= \lambda_H$ (means it is possible to send data to one of the neighbouring H.) **(scenario 4)**. So, the local H will check for an Idle neighbour.

Calculate the response time in H (t_H) or cloud (t_{cloud}) based on the state of the neighbour:

1- [0,0] $\rightarrow$ Both neighbours are busy, then the H offloads the data to the Cloud **(Scenario 5)**.

2- [1,0] $\rightarrow$ neighbour 1 is idle, then the H offloads the data to the H (neighbour 1) **(Scenario 4)**.

3- [0,1] $\rightarrow$ neighbour 2 is idle, then the H offloads the data to the H (neighbour 2) **(Scenario 4)**.

4- [1,1] $\rightarrow$ Both neighbours are idle, then the local H offloads the data to the nearest H (neighbour) **(Scenario 4)**.

Case 2: $\lambda_z > \lambda_H$ (means the data must be offloaded to the Cloud.) **(scenario 5)**.

Calculate: $\lambda_{Cloud} \leftarrow \lambda_z$

λ_{Cloud} is H's workload which needs to be offloaded to the cloud.

Calculate: the response time in the Cloud:

$$t_{cloud} \leftarrow \frac{\lambda_{Cloud}}{\mu_{Cloud}}$$

4- Calculate the delay for each patient t_{pi} according to Equation 3.9
5- Finally, calculate the system SRT: $SRT \leftarrow \frac{\sum_{1}^{Ps} t_{pi}}{Ps}$

3.4 HMAN system model

The HMAN architecture is cased studied on the healthcare system in the UK for a more convenient derivation of the proposed model. The healthcare system in the UK consists of two main levels, namely, the general practices (GPs) level and the general Hs (GHs) level. The GPs are the primary MCs distributed throughout the city and are responsible for providing medical services for local patients. Meanwhile, the GHs are the general Hs, and each one is responsible for hospitalising patients referred by a group of GPs.

In terms of applying the HMAN architecture on the healthcare infrastructure in the UK, it can be represented as an undirected graph G = {P ∪ GP ∪ GH, E}, where P is a set of monitored patients, $P = \{p_1, p_2, p_3, \ldots, p_i\}$, GP is a set of MCs in a local area of the city, $GP = \{gp_1, gp_2, gp_3, \ldots, gp_j\}$, GH is a set of the general Hs in the city, $GH = \{gh_1, gh_2, gh_3, \ldots, gh_z\}$, and E represents the connection links that can be a wire (optical fibre) or wireless (Wi-Fi connection). Figure 3.4 illustrates the components of the HMAN system.

In the HMAN system, the patient can access any local GP service either through a direct link if he/she is in that GP coverage area or through the access point (AP) he/she is connected to. Each GP is connected to two GPs (the two nearest neighbours) and is responsible for providing services to patients within its area of responsibility. Meanwhile, every GH is connected to two GHs (the two nearest neighbours) and is responsible for all local GPs located within its area of responsibility. The fibre

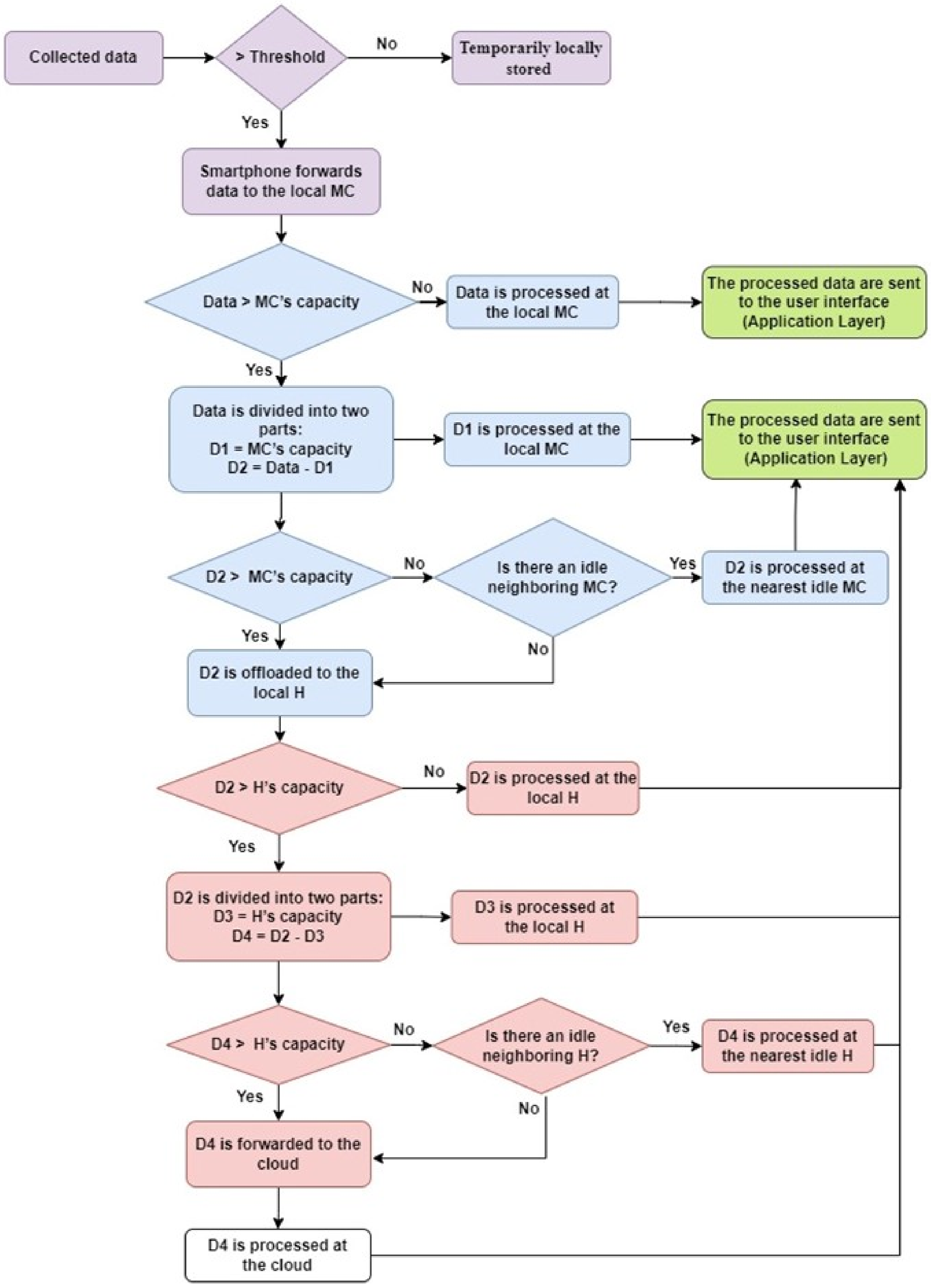

Collected data
> Threshold
No
Temporarily locally stored
Yes
Smartphone forwards data to the local MC
Data > MC's capacity
No
Data is processed at the local MC
The processed data are sent to the user interface (Application Layer)
Yes
Data is divided into two parts: D1 = MC's capacity D2 = Data - D1
D1 is processed at the local MC
The processed data are sent to the user interface (Application Layer)
D2 > MC's capacity
No
Is there an idle neighboring MC?
Yes
D2 is processed at the nearest idle MC
Yes
No
D2 is offloaded to the local H
D2 > H's capacity
No
D2 is processed at the local H
Yes
D2 is divided into two parts: D3 = H's capacity D4 = D2 - D3
D3 is processed at the local H
D4 > H's capacity
No
Is there an idle neighboring H?
Yes
D4 is processed at the nearest idle H
Yes
No
D4 is forwarded to the cloud
D4 is processed at the cloud

Figure 3.2: HOSSC data flow and offloading algorithm of the HMAN architecture.

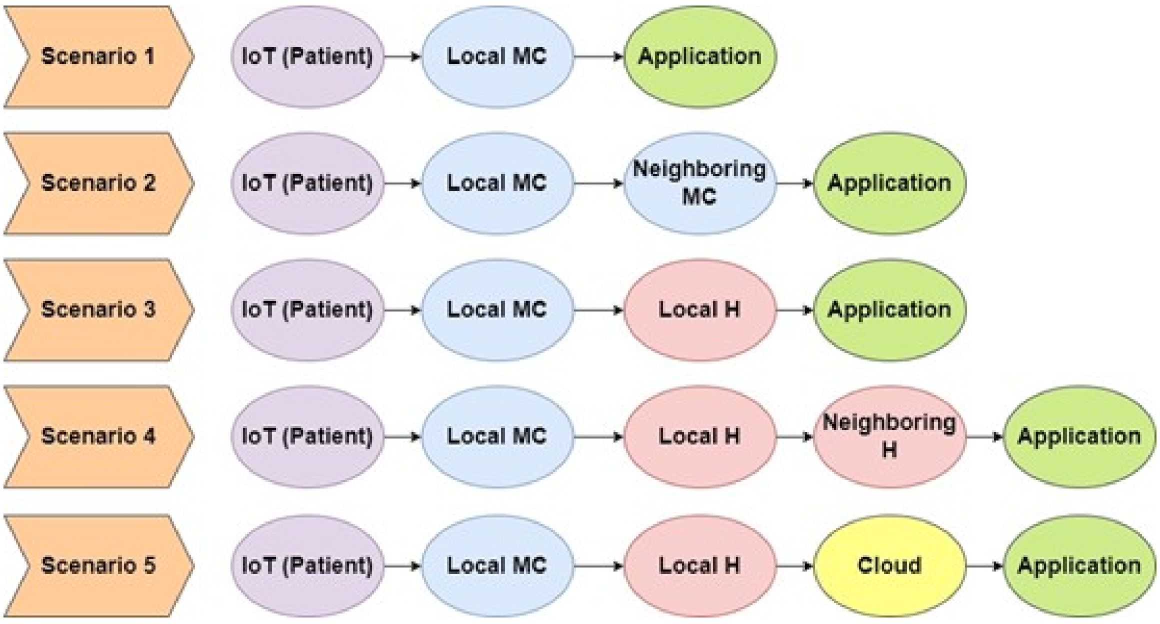


Figure 3.3: All potential scenarios in the proposed system.

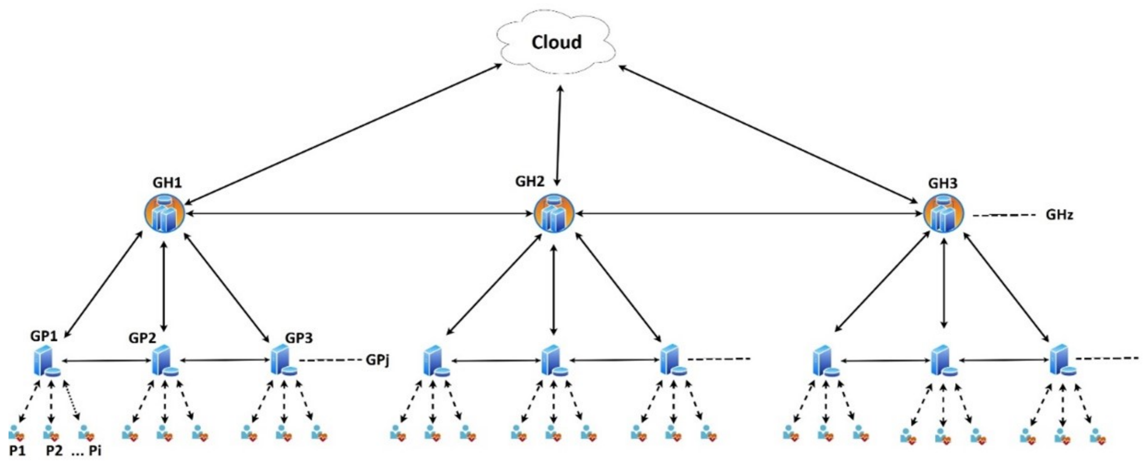


Figure 3.4: All Components of the HMAN system. The arrow directions indicate the flow of data. P = patients, GP = medical centres (general practice), GH = general hospitals. The GPs and GHs are hierarchically connected.

optics are utilised to link the aforementioned healthcare facilities to provide high-speed connections. In addition, the remote cloud is accessible to all GHs through high-speed Internet connections.

In the first layer of the architecture, the patient's smartphone classifies the collected data into normal or abnormal according to a threshold value of each health issue. The normal data are invisible to the further layers in the system, and the only offloaded data to the local GP for processing are the abnormal data. Accordingly, each patient, pi, has a task that randomly arrives in the system with an arrival rate λ_i, according to the Poisson process.

Figure 3.5 depicts the system model where the data can be offloaded from a patient to a local GP either through a direct connection with a wireless delay (D_{wi}) or through an AP with a total delay of ($D_{wi} + D_{AP}$). In addition, V, S and B denote the delay in between GPs, GPs to GHs or in between GHs and the internet delay, respectively.

3.5 Offloading system model

The HMAN system is designed as a queuing model. The offloaded data can be processed at the GPs, GHs, or remote cloud according to the servers' status in each site. This study assumes that all GPs' servers are identical, and the same assumption applies on GHs' servers. However, the GHs host more servers with higher specifications and capabilities than the ones hosted at the GPs. Let us denote the number of servers at each GP site as n and the number of servers at every GH as m. In addition, GP and GH sites are modelled using M/M/n and M/M/m queue models, respectively, to accurately reflect their service operations. In these models, each GP or GH comprises n or m homogeneous servers with fixed service rates of µGP and µGH. The M/M/n and M/M/m models are chosen because they effectively capture the nature of GP and

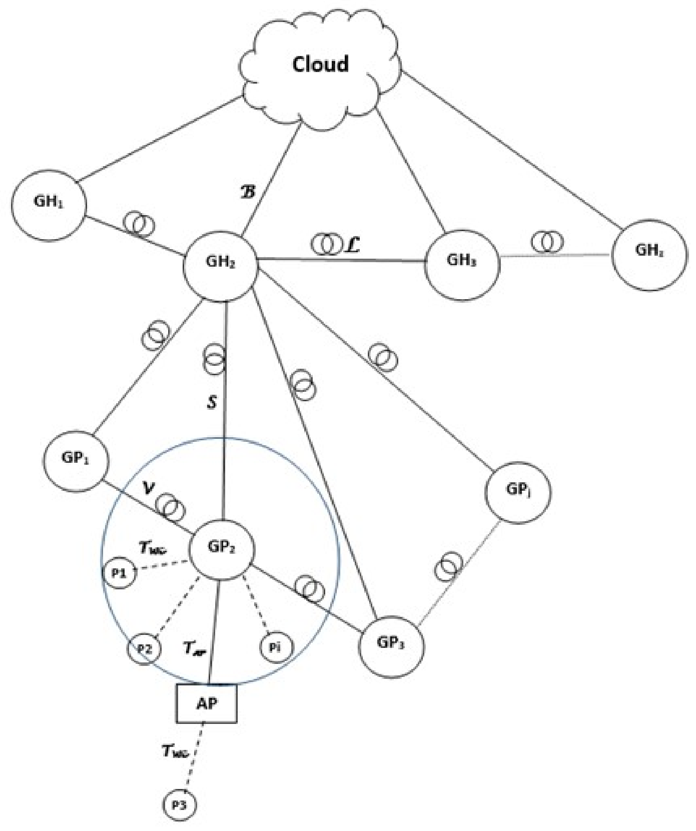


Figure 3.5: HMAN system model.

GH services, where multiple servers handle patient data and requests simultaneously. These models assume exponential inter-arrival and service times, which align well with the stochastic nature of healthcare service requests and processing times [131]. The following part presents the offloading system model after applying the data flow scenarios described in Section 3.3 on the case studied healthcare system in the UK.

The collected data from each monitored patient create a task that can be executed at the patient's smartphone when the data are determined to be normal; at the local GP or its neighbouring GPs when the data are determined to be abnormal; at the local GH or its neighbouring GHs when the abnormal data require more processing capacity than the cooperative GPs' capacity; or at the cloud when the previous units cannot afford enough processing capacity to execute the abnormal data.

The average queuing time, f_Q, can be found using the equation below [132]:

$$f_Q(\lambda) = \frac{C(n, \lambda/\mu)}{n\mu - \lambda} \tag{3.2}$$

Where λ is the arrival rate, and $C(n, A)$ is Erlang's formula [132] obtained by:

$$C(n, A) = \frac{(\frac{(nA)^n}{n!})(\frac{1}{1-A})}{\sum_{k=0}^{n-1}(\frac{(nA)^k}{k!}) + (\frac{(nA)^n}{n!})(\frac{1}{1-A})} \tag{3.3}$$

3.5.1 Response Time at GPs

Assuming that $P = \{p_1, p_2, p_3, \ldots, p_i\}$ is a set of patients assigned to the local GP, the patients' smartphones offload the collected abnormal data to the local GP for processing. If the local GP is overloaded, then the GP processes the data within its available capacity, whilst the remaining data are offloaded either to the nearest idle GP or to the local GH based on the criterion, $\lambda_{GPmax} < (\lambda_{GPsum} = \sum \lambda_i)$, where λ_{GPmax} represents the maximum GP workload, and λ_{GPsum} is the arrival workloads from the patients.

To determine the amount of data to be processed at the local GPs and those to be offloaded to the cooperative units, the calculation of the data fraction scalar $K1$ is required as follows:

$$K_1 = \begin{cases} 1, & \text{if} \quad \lambda_{GPmax} \geq \lambda_{GPsum} \\ \frac{\lambda_{GPmax}}{\lambda_{GPsum}}, & \text{otherwise.} \end{cases} \tag{3.4}$$

Then, the response time at the GPs, t_{GP}, for the received data can be found as follows:

$$t_{GP} = f_Q(K_1.\lambda_{GPsum}) + \frac{K_1.\lambda_{GPsum}}{\mu_{GP}} \tag{3.5}$$

Where $f_Q(K_1.\lambda_{GPsum})$ represents the queuing time needed for the fraction of the arrived data to be processed at the GP, and $\frac{K_1.\lambda_{GPsum}}{\mu_{GP}}$ represents the service time at the GP for the processed workload.

3.5.2 Response Time at GHs

Assuming that $GP = \{gp_1, gp_2, gp_3, \ldots, gp_i\}$ is a set of GPs assigned to the local GH, the GP offloads its data to the local GH. In case the local GH is overloaded, the data will be offloaded either to the nearest idle GH or to the cloud according to: $\lambda_{GHmax} < (\lambda_{GHsum} = \lambda_{GPsum} - \lambda_{GPmax})$, where λ_{GHmax} represents the maximum GH workload, and $(\lambda_{GHsum}$ is the arrival workloads from the GHs.

To determine the amount of data to be processed at the local GH and those to be offloaded to the neighbouring GHs or further to the cloud, the calculation of the data fraction scalar $K2$ is required as follows:

$$K_2 = \begin{cases} 1, & \text{if} \quad \lambda_{GHmax} \geq \lambda_{GHsum} \\ \frac{\lambda_{GHmax}}{\lambda_{GHsum}}, & \text{otherwise.} \end{cases} \tag{3.6}$$

Accordingly, the response time at the GHs, t_{GH}, for the arrived data can be found as follows:

$$t_{GH} = f_Q(K_2.\lambda_{GHsum}) + \frac{K_2.\lambda_{GHsum}}{\mu_{GH}} \tag{3.7}$$

Where $f_Q(K_2.\lambda_{GHsum})$ represents the queuing time needed for the fraction of arrived data to be processed at the GH, and $\frac{K_2.\lambda_{GHsum}}{\mu_{GH}}$ denotes the service time at the GH for the processed workload.

3.5.3 Response Time at the cloud

The cloud can be modelled as M/M/∞ queue referring to its unlimited resources. Thus, the queuing time at the cloud can be considered as zero, and the response time at the cloud can be estimated as follows:

$$t_{Cloud} = \frac{\lambda_{cloud}}{\mu_{GH}} \tag{3.8}$$

According to Equations 3.5, 3.7 and 3.8, the average response time of the offloaded tasks by a patient in the HMAN system is calculated as follows:

$$\begin{split} t_{pi} = T_{wi} + E_{AP} * T_{AP} + (c_1 + 1) * t_{GP} + c_1 * V + c_2 * S \\ + c_2 * t_{GH} + c_3 * L + c_3 * t_{GH} + c_4 * B + c_4 * t_{Cloud} \end{split} \tag{3.9}$$

Where c1, c2, c3 and c4 represent the counters to count the number of times to reach a certain unit in the system; $c1$ = GP neighbours' counter; $c2$ = the local GH counter; $c3$ = GH neighbours' counter; and $c4$ = the cloud counter. E_{AP} represents the link between the patient and the local GP; and $E_{AP} = 0$ (direct connection) or 1 (through an AP). Finally, the system response time (SRT), which represents the average system latency required to deliver patient data to one of the health facilities within the architecture, can be determined by:

$$SRT = \frac{\sum_{i=1}^{n} t_{pi}}{n} \tag{3.10}$$

3.6 Results

This section presents the simulation results of all the conducted potential scenarios according to the HMAN system model. A few parameters are defined and listed in Table 3.1 for a more convenient evaluation of the system's feasibility and advantages. The SRT when simultaneously processing varying numbers of patients sending data is considered a metric to measure the performance of the system. Meanwhile, the other important system aspect to look for is the system scalability and how far the HMAN system can sustain a reliable service responding to the simultaneous data flow.

Table 3.1: System parameters

Symbol	**Parameter**	**Value**
Ps	Number of patients	100, 200, 300
n	Number of servers in each MC (GP)	5, 6, 7
m	Number of servers in each GH	10, 12, 14
r_{Wi}	Link rate between IoT device and AP	54 Mbps
r_{AP}	link rate between AP and the local GP	100 Mbps
r_{Fib}	link rate between the GPs and GHs	Up to 10 Gbps
λ_i	Packet size	30 KB
T_{wi}	Wireless Delay	4 ms
T_{AP}	Delay between AP and the local GP	2 ms
V	Delay between two neighboring GPs	$\frac{(1-K_1).\lambda_{GPsum}}{10Gbps}$
S	Delay between the GPs and GHs	$\frac{K_2.\lambda_{GHsum}}{10Gbps}$
L	Delay between two neighboring GHs	$\frac{(1-K_2).\lambda_{GHsum}}{10Gbps}$
B	Internet Delay	20 ms
μ_{GP}, μ_{GH}	Each GP, GH server service rate	100,200 KB per ms
μ_{Cloud}	Cloud service rate	1000 KB per ms
λ_{GPmax}	Maximum GP workload	200×n KB
λ_{GHmax}	Maximum GH workload	200×n KB

The cooperative units (i.e., the local and neighbouring GPs and GHs) are gradually deployed to respond to different workloads to examine how efficient and scalable the hierarchy HMAN architecture is. The system units are gradually engaged in five stages starting from only the local GP and ending at the whole system, including all the cooperative units.

Initially, the SRT to the data from a certain workload that can only be processed at the local GP and Cloud is evaluated for greater clarity in visualising this progressive deployment. Then, more units are gradually deployed in between the local GP and the cloud for the same workload. The neighbouring GPs, the local GH, the neighbouring GHs, and the whole cooperative system are engaged to respond to the same workload under these stages of more involved units. This situation can emphasise how the hierarchy architecture can collaboratively manage the workload. In terms of workloads, the system is tested for 100, 200 and 300 patients, and the results of the SRT are presented in Figures 6, 7 and 8 respectively. Each figure includes five graphs representing the SRT of the HMAN system under each of the aforementioned five stages for the same workload. For a fair evaluation of the suggested architecture in responding to different workloads, each of the five graphs in Figure 6 is discussed with the corresponding graphs from Figures 7 and 8. Figure 6 shows that the SRT is reduced when the data are offloaded from the local units (GPs or GHs) to their neighbouring units rather than being transmitted to the upper layers.

Figure 3.6a shows that when two layers and 100 patients are sending data at the same time, a portion of the data must be offloaded to the cloud due to the limited resources at the GP layer. Accordingly, the response time is longer, which is undesirable in many pathological situations where a faster response is required. Additionally, these offloaded data are sufficient to raise privacy concerns because a portion of the data are sent outside the network in such a scenario. Furthermore, this situation can be even more challenging when serving more patients (i.e., a large number of data are likely to be processed outside the network), as shown in Figures 3.7a and 3.8a for 200 and 300 patients, respectively.

In the second stage, the system supports the local GP by two GP neighbours. The local GP first checks if the data can be offloaded to the nearest available neighbour (i.e. when the overload data are less than λ_{GPmax}) rather than being sent to the cloud. Such neighbouring GP involvement creates an additional scenario to locally process the data and achieve more privacy, greater inherited computational capacity and less latency. As a result, a reduced SRT is obtained, and the system is expanded to ensure that more patients can be served within the local network, as illustrated in Figures 3.6b, 3.7b, and 3.8b.

The following stage is to engage the local GH to serve as a reference point for all the GPs in the same area. Accordingly, an additional scenario to process the data is provided compared with the previous stage. When the data exceed the capacity of the local GP neighbours, this new scenario becomes valid, and the overload data are offloaded to the local GH. Although the SRT increased in responding to a 100 patients' workload (Figure 3.6c), the SRT decreased when handling heavier loads of 200 and 300 patients (Figures 3.7c and 3.8c). The higher SRT in a 100-patient workload is attributed to the longer distance the offloaded data can go through as it travels from the local GP to the local GH, which is often located further than the neighbouring GPs. Nevertheless, the computational capacity of the GHs is higher than the GPs; hence, a larger number of patients can be accommodated. Furthermore, a higher level of the patients' privacy is achieved because this system expansion still assures the local processing of the data.

When the local GH is connected to the two GH neighbours in the fourth stage, the system's performance is improved. In the previous stage, when the data overload the local GP and GH, the overload data must be offloaded to the cloud, exposing the data to less privacy and higher latency. Meanwhile, this stage introduces an additional scenario to locally process the data at the neighbouring GHs. The local GH checks the status and the capacity of the neighbouring GHs and decides whether to send the overload data to the neighbouring GHs or to the cloud.

Consequently, the system would have more probability of local data handling and less possibility of reaching the cloud. Thus, more privacy is met, more patients are served, and less latency can be achieved. The results depicted in Figures 3.6d and 3.7d cannot show this SRT improvement because the local GH along with the previous units are capable of processing the data; hence, no overload data reaches the local GH neighbours. However, the aimed SRT improvement can be clearly seen in Figure 3.8d because the system affords further resources to avoid the data offloading to the cloud.

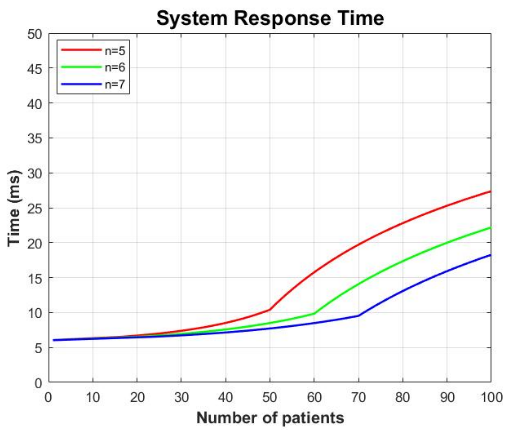
In the final stage, when the entire system is connected, more computational capacity is gained, resulting in a more powerful processing platform capable of responding to a higher number of monitored patients. All the five scenarios conducted in this study are involved in this stage. The results presented in Figures 3.6e, 3.7e and 3.8e show that the performance is significantly improved compared with those in the previous stages. Moreover, the more healthcare units are involved in the cooperative HMAN architecture, the more computational capacity can be inherited, thereby reflecting less SRT. Furthermore, the patients' privacy is also increased because such a collaborative network achieves a higher probability that data are locally processed.

The simulation considered the highest expected values of the time delay to calculate. In addition, all patients are assumed to be indirectly connected to the health centres; hence, the T_{AP} of 2 ms was added to all the considered scenarios. Accordingly, the obtained results are for the worst case when it comes to the consideration of the time delay assumptions. Nevertheless, the presented HMAN framework efficiently serves a workload of 300 patients simultaneously requesting service with a latency of 31.45 ms when compared the latency of 75 ms achieved in [69]. This remarkable reduction in the service delay is attributed to the presented architecture that utilises the nearest units for cooperative data processing. This utilisation of the MCs' and Hs' geographical locations shortened the distance between patients and health facilities and eventually reflected reduced latency.

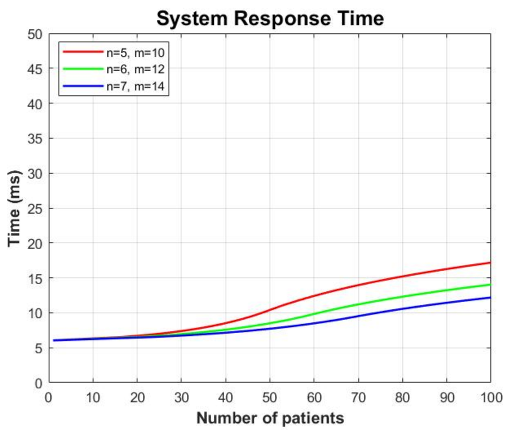
Furthermore, the system performance is improved in all resulting graphs when increasing the servers at the GPs (n) from 5 and 6 to 7 and the GHs (m) from 10 and 12 to 14. This result makes sense due to the increase in resources.

Consequently, more data are locally processed rather than being offloaded to the upper layers. Table 3.2 illustrates the system achievements to indicate the performance in responding to different workloads as we gradually engage more units in the five testing stages. The aforementioned table emphasises how data become less exposed to the cloud despite the increased number of patients from 100 to 300 as more units are gradually involved from stage 1 utilising only the local GP and the cloud to the final stage, including all the cooperative units. For example, in responding to a 300-patient workload with n = 6 and m = 12, moving from stage 1 to stage 5, the numbers of patients who reached the cloud are 240, 180, 60, 0 and 0. Specifically, a ubiquitous and scalable health system with high service availability, more computing capacity, less latency, and better privacy is obtained.

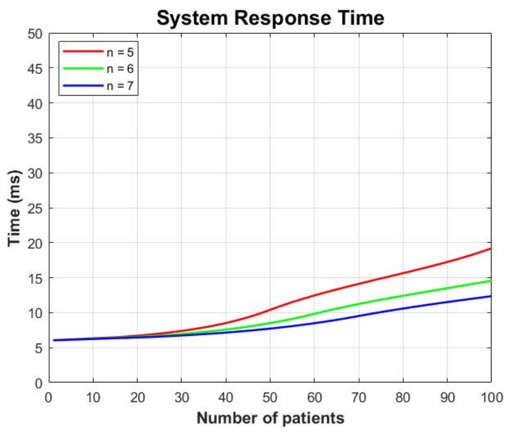
It is worth noting that in the context of healthcare, the significance of rapid response times, even within the range of milliseconds, cannot be overstated. While millisecond differences may be critical only in applications such as virtual reality or gaming, healthcare scenarios equally benefit from such responsiveness. For instance, in emergencies such as remote patient monitoring for heart attack or stroke symptoms, every millisecond of latency can be crucial in initiating timely interventions. Rapid response times ensure that alerts are processed and communicated almost instantaneously, thereby expediting the delivery of emergency medical services and potentially saving lives. Furthermore, in telemedicine applications where real-time data exchange between patients and healthcare providers is essential, lower latency enhances the quality of interaction and decision-making processes. Additionally, certain medical devices, such as insulin pumps and other automated drug delivery systems, depend on precise and timely data processing to maintain patient safety and treatment efficacy. Therefore, achieving response times in the millisecond range is paramount for enhancing the overall effectiveness and reliability of modern healthcare systems.



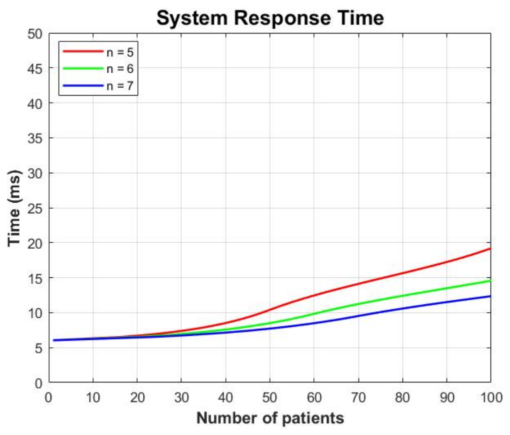
(a)



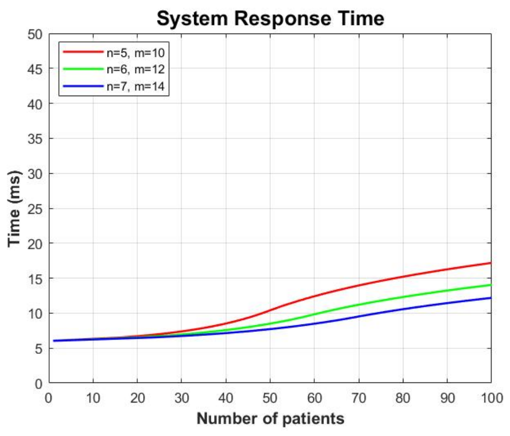
(b)



(c)

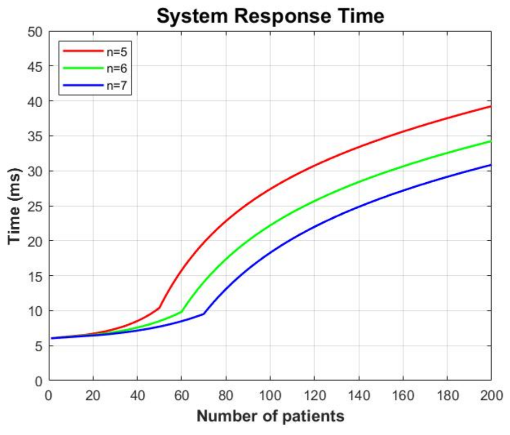


(d)

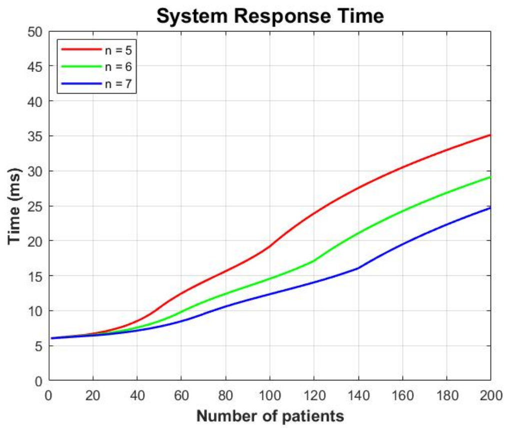


(e)

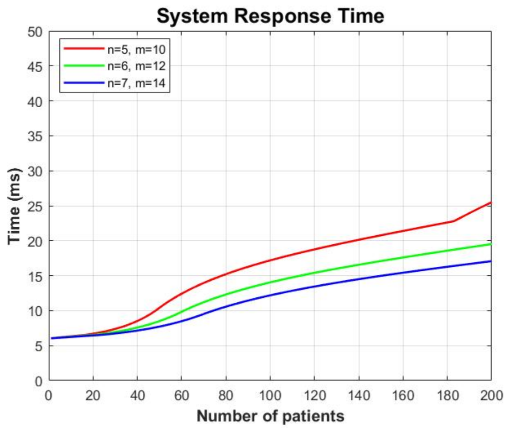
Figure 3.6: SRT of 100 patients in the expected different scenarios. (a) Patient $\rightarrow$ Local general practice (GP) $\rightarrow$ Cloud. (b) Patient $\rightarrow$ Local GP $\rightarrow$ GP neighbour $\rightarrow$ Cloud. (c) Patient $\rightarrow$ Local GP $\rightarrow$ Local general H (GH) $\rightarrow$ Cloud. (d) Patient $\rightarrow$ Local GP $\rightarrow$ Local GH $\rightarrow$ GH neighbour $\rightarrow$ Cloud. (e) Whole system



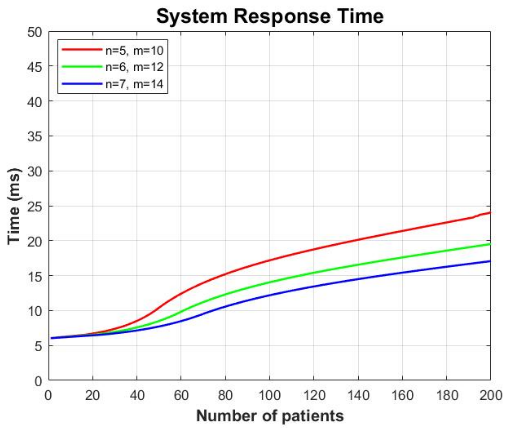
(a)



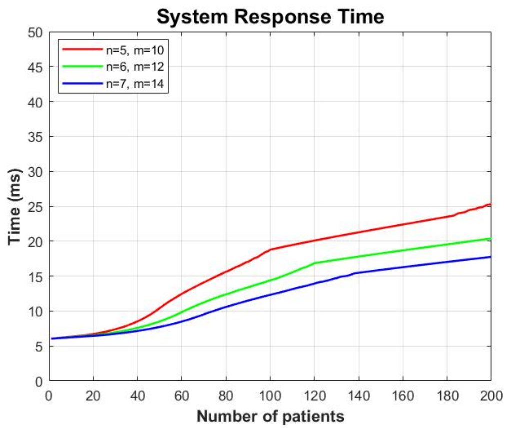
(b)



(c)

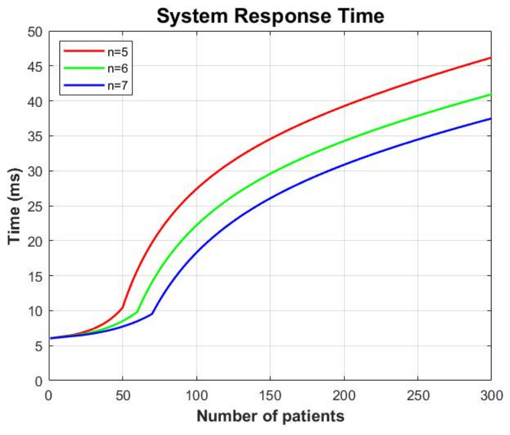


(d)

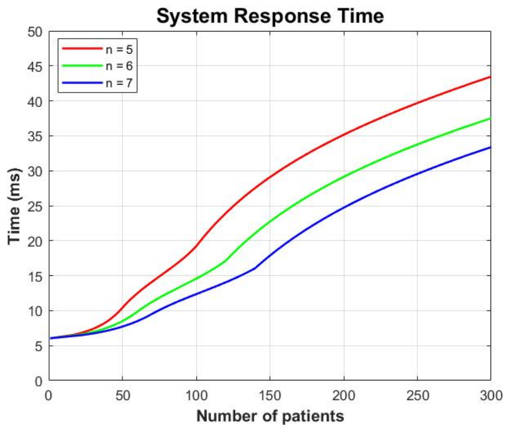


(e)

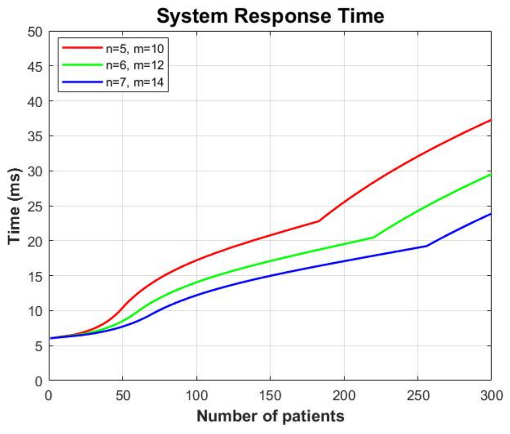
Figure 3.7: SRT of 200 patients in the expected different scenarios. (a) Patient $\rightarrow$ Local general practice (GP) $\rightarrow$ Cloud. (b) Patient $\rightarrow$ Local GP $\rightarrow$ GP neighbour $\rightarrow$ Cloud. (c) Patient $\rightarrow$ Local GP $\rightarrow$ Local general H (GH) $\rightarrow$ Cloud. (d) Patient $\rightarrow$ Local GP $\rightarrow$ Local GH $\rightarrow$ GH neighbour $\rightarrow$ Cloud. (e) Whole system



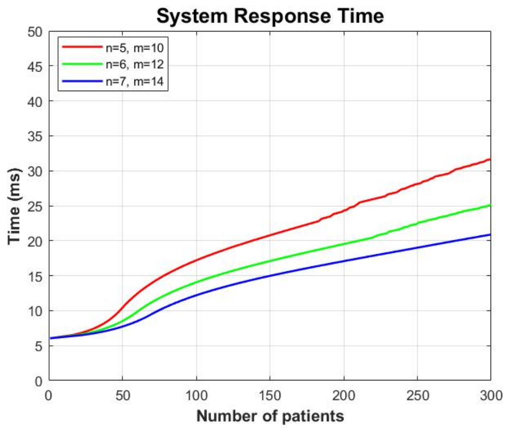
(a)



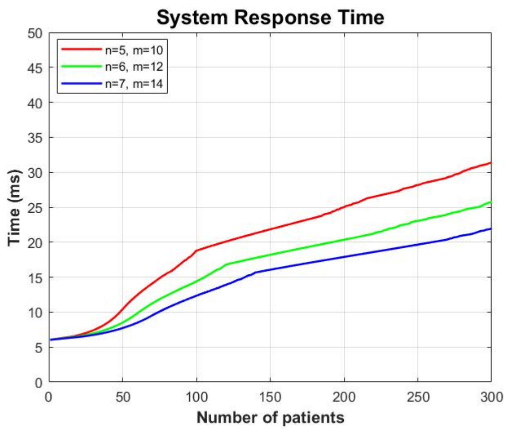
(b)



(c)



(d)



(e)

Figure 3.8: SRT of 300 patients in the expected different scenarios. (a) Patient → Local general practice (GP) → Cloud. (b) Patient → Local GP → GP neighbour → Cloud. (c) Patient → Local GP → Local general H (GH) → Cloud. (d) Patient → Local GP → Local GH → GH neighbour → Cloud. (e) Whole system

Table 3.2: Number of patients served in each architecture unit at every deployment stage (purple indicates unused units at a certain stage)

Stage	No. of patients	No. patients served at														
		Local GP			GP neighbours			Local GH			GH neighbours			Cloud		
		n = 5	n = 6	n = 7	n = 5	n = 6	n = 7	m = 10	m = 12	m = 14	m = 10	m = 12	m = 14	n = 5 m = 10	n = 6 m = 12	n = 7 m = 14
1	100	50	60	70										50	40	30
	200	50	60	70										150	140	130
	300	50	60	70										250	240	230
2	100	50	60	70	50	40	30							0	0	0
	200	50	60	70	50	60	70							100	80	60
	300	50	60	70	50	60	70							200	180	160
3	100	50	60	70				50	40	30				0	0	0
	200	50	60	70				133	140	130				17	0	0
	300	50	60	70				133	160	187				117	80	43
4	100	50	60	70				50	40	30	0	0	0	0	0	0
	200	50	60	70				133	140	130	17	0	0	0	0	0
	300	50	60	70				133	160	187	117	80	43	0	0	0
5	100	50	60	70	50	40	30	0	0	0	0	0	0	0	0	0
	200	50	60	70	0	0	0	133	140	130	17	0	0	0	0	0
	300	50	60	70	0	0	0	133	160	187	117	80	43	0	0	0

3.7 Limitations and further improvements

Finally, this study has some limitations that the developed system anticipates data equally without taking urgency into account, which causes delays in the treatment of critically ill patients. By classifying the patient's data into multiple classes according to the patient's condition at edge servers, it would be easier to identify the most severe patients. Thus, giving them priority in receiving services, whether sending an ambulance, preparing necessary medical staff, etc. Therefore, the main objective of our next study is to improve this work to build an intelligent healthcare monitoring system that can dynamically regulate patient data flow based on each patient's individual health status.

3.8 Summary

This wok proposed a cooperative hierarchical healthcare architecture, named HMAN, with Edge/Fog computing. The suggested architecture is supported by the HOSSC algorithm that provides five offloading and processing scenarios and facilitates SRT calculating. The architecture consists of four layers, namely, IoT, Edge/Fog, cloud and application layers. Model analysis and thorough experimentation revealed the benefits of this architecture. The research confirmed the soundness of the suggested architecture and provided a method for measuring the system performance under various network workloads. This study investigated various scenarios to ensure that the data are locally processed to meet the objectives of more privacy and less latency. Five different stages have been established to aid in the evaluation of the presented architecture. The HMAN architecture achieved a robust and scalable healthcare system exploiting the existing infrastructure in the city with low latency, ranging from 6.043 to 31.45 ms, considering the various workloads. According to the results, the hierarchical architecture achieved inherited computational capacity, higher system scalability and greater availability. Moreover, the HMAN architecture assured higher probability of local data processing in addition to the achieved less latency and higher privacy. The findings clearly showed that the suggested HMAN architecture is efficient enough for use in the healthcare domain. Nevertheless, some challenges in realising such architecture can be anticipated to include the prioritisation of the critical cases of some patients to provide an exceptional allocation of the healthcare resources for them. Finally, this study can be extended by utilising modern machine learning approaches (e.g., deep learning) in recognizing diverse traffic and determining the optimal placement for GP servers in a certain area to improve the performance and reduce the deployment cost.

Chapter 4

Optimal Intelligent Edge-Servers Placement in the Healthcare Field

4.1 Introduction

The importance of healthcare has increased worldwide due to its direct relationship with the quality of human life. An effective and robust healthcare system leads to a strong and confident society. However, governments and the public sectors face many challenges due to the rising number of patients year after year, as well as growing epidemiological concerns. These challenges become more serious with the elderly population requiring constant monitoring. As a result, it is becoming increasingly difficult for traditional healthcare systems, which require one-to-one contact between the caregiver and patient, to expand to accommodate the growing patient population [133]. Moreover, the financial burden brought on by administrative and operational expenses has a propensity to lead to system burnout. Thus, smart healthcare (s-health) systems are becoming increasingly necessary to meet these requirements. Furthermore, finding smart, cost-efficient systems that have the ability to remotely monitor and supervise patients is a significant advancement in medical science, offering reasonable solutions to both current and future problems. These systems should provide high-quality medical care at a limited cost to guarantee their long-term viability [134].

S-health is anticipated to make a substantial contribution towards reducing hospitalisation rates and delivering telehealth services to remote patients at a reasonable cost. This is along with other advantages, such as accuracy, scalability, energy efficiency, configurability, and maintainability. However, the question is how to switch from conventional healthcare systems to s-health systems [135].

The transition of conventional healthcare practices to s-health has been expedited by advancements in computational intelligence, mobile communication technologies, and Internet of Things (IoT) technologies [136]. The development of artificial intelligence (AI) will have a great impact by enhancing the intelligence, autonomy, dynamic nature, and adaptability of healthcare systems. These services are predicted to revolutionise healthcare by expediting diagnosis and treatment procedures, lowering the cost of doctor visits, and improving the standard of patient care.

Moreover, combining edge computing with smart health is so versatile that it can provide short delay, save network bandwidth, lower power consumption, as well as deliver security and data privacy benefits [136]. The strategic placement of edge-servers plays a crucial role in achieving these advantages, including reduced deployment cost, minimal latency, load balance, etc. That is, the question of where edge-servers should be placed within a network must be considered. Moreover, the edge-servers placement problem becomes much more important when taking into account the utilisation of edge-servers in a wireless local area network (WLAN).

This chapter aims to explore the potential of using AI in conjunction with edge computing to enhance healthcare systems and make them more sophisticated. Firstly, an intelligent priority mechanism is proposed to identify critical patients who require urgent medical assistance. The researcher builds upon the prior work, the HMAN environment published in [91], as a baseline to achieve better Quality of Service (QoS) and Quality of Experience (QoE) when utilising AI with edge/fog computing in the healthcare sector. The HMAN system treated all data equally, without considering urgency, which could cause delays in treating critically ill patients. By categorising users' data into different classes, based on patients' conditions, it becomes easier to identify and prioritise the most severe cases, enabling them to receive prompt and appropriate services such as sending an ambulance or preparing necessary medical staff. The ultimate objective of this study is to develop intelligent monitoring and healthcare systems for medical facilities that can dynamically regulate patient flow based on each individual health status.

Secondly, strategic placement of edge-servers in the HMAN environment can significantly reduce deployment costs and minimise the overall delay between patients and edge-servers, thus enhancing the performance of healthcare applications. In regions where multiple MCs exist, it is not necessary to deploy edge-servers at each MC due to financial limitations. However, it is essential to ensure that the delay requirements for each task are met. Since all MCs are identical, any MC can be a potential location for placing edge-servers. Nonetheless, a subset of the MCs can be selected to reduce the deployment budget while maintaining an acceptable level of delay. The question that arises now is how to select the best MCs subset to achieve the research objectives.

As human life takes precedence in research pertaining to the design and proposal of healthcare systems, the top research priority is to ensure that health services arrive as quickly as possible with minimal cost. The key objective is to design an intelligent healthcare system while reducing the cost of server deployment and maintaining acceptable latency. The following is a summary of the contributions of this study:

1. A priority processing/offloading mechanism based on AI is introduced, aimed at identifying critical patients who require urgent medical assistance, thereby enhancing QoS and QoE.

2. The edge-server placement problem is investigated and an optimal edge-servers placement (OESP) algorithm is proposed. This algorithm aims to achieve cost-efficient architecture with lower delay and comprehensive coverage. The main objective is to deploy edge-servers to subsets of MCs within a given area, providing edge health services to monitored patients.

3. Simulations are conducted to evaluate the performance of the proposed algorithms. The results demonstrate favourable outcomes, showcasing a cost-effective system with lower latency when compared to existing algorithms in the literature.

Chapter Organisation: The remainder of this chapter is organised as follows: Section 4.2 presents the proposed priority mechanism; Section 4.3 introduces the optimal edge-servers problem formulation; Sections 4.4 describes the proposed optimal edge-servers placement (OESP); Section 4.5 presents and discusses the simulation results; and finally, Section 4.6 presents a summary of the study.

4.2 Priority mechanism based on AI

This work involved the creation of an intelligent healthcare system for data classification of patients, which would increase data collection effectiveness and streamline processing. By prioritising data based on the urgency of patients, better Quality of Service (QoS) and Quality of Experience (QoE) can be achieved. The proposed model, illustrated in Figure 4.1, comprises two main levels: the patient level (IoT layer) and the local MC level (Edge/Fog Layer).

The patient level (a smartphone or watch) has two functions: data collection and filtering (preprocessing). At the local MC level, the arriving data is processed through three steps: classification, prioritisation, and decision-making.

At the patient level, a wearable device (smartphone or watch) collects data and

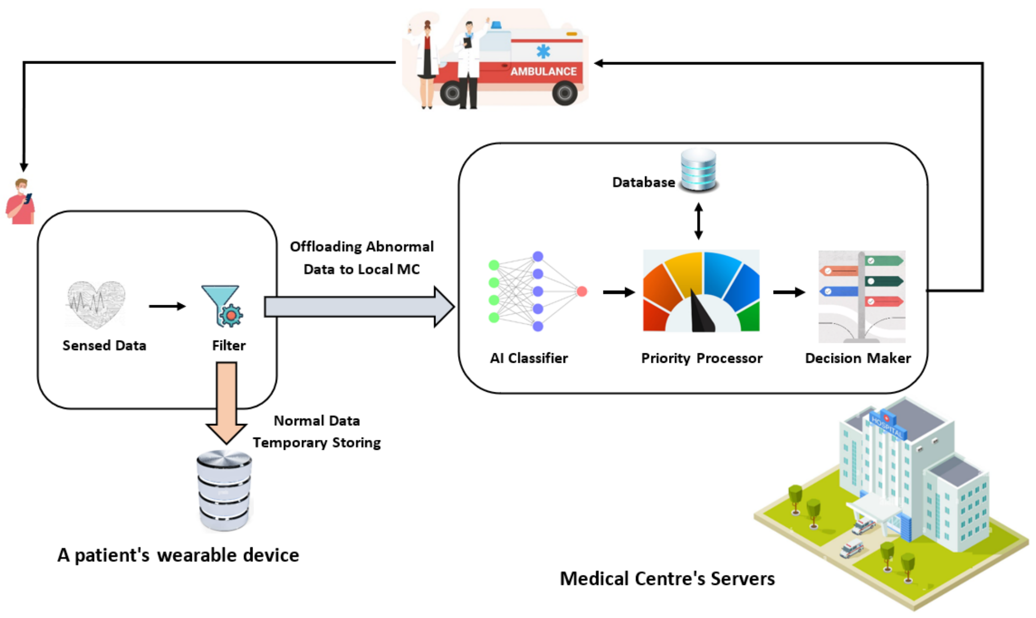


Figure 4.1: Model of the proposed priority mechanism.

performs initial data preprocessing and analysis (i.e., aggregation, fusion, filtering, and classification). As a result, based on a predetermined threshold assigned according to the patient's condition, the collected data are classified as normal or abnormal. While the abnormal data is offloaded to the next layer for additional actions, the normal classified data are temporarily stored locally without requiring any further action.

At the local MC level, the received data undergoes three steps to determine the appropriate course of action. Firstly, an AI method, such as machine learning (ML) or deep learning (DL), is employed to classify the data into multiple categories enabling the determination of the level of risk associated with each patient. Subsequently, a priority processor assigns a priority level to each patient based on their historical record, which is stored in a database. Finally, a decision is made regarding whether the data should be processed locally or offloaded to other units based on their priority levels. For example, if the local MC receives a simultaneous influx of 100 patients, but can only provide medical services for 50, it prioritises the first 50 patients on the list, while the remaining patients are offloaded to other units within the architecture using the HOSSC algorithm outlined in chapter 3. Algorithm 1 represents the proposed priority processing/offloading scenario, which dynamically updates patient lists based on their priority levels. This ensures that the most urgent cases receive medical attention at the local MC, while other cases are handled by other units in the architecture.

Algorithm 1: Priority processing/offloading algorithm

```
Input:$\lambda_{sum}, \lambda_{max}$
Output: Priority mapping patient offloading
For a given situation at a particular time instant t the following will be done.
If $\lambda_{sum} \leq \lambda_{max}$
  All patients can be served by the local MC
else      // The priority mechanism will be turned on
  $\lambda_{sum}$ is classified into classes by the AI classifier
  Class A $\rightarrow$ Priority = 0   // Normal cases, Discarded
  Class B $\rightarrow$ Priority = 1   // Mild case
  Class C $\rightarrow$ Priority = 2   // Moderate case
  Class D $\rightarrow$ Priority = 3   // Severe case
  .... // (if needed depending on the type of disease)
end if statement
SortedPatientsMatrix = [a high_priority_patient ... a less_priority_patient]
Then:
The Local MC will serve patients based on their order in SortedPatientsMatrix
```

4.3 Optimal edge-servers problem formulation

Our goal is to effectively deploy edge-servers to specific MCs within a region to meet the needs of all patients under monitoring. The area is represented as a twodimensional space (grid) in which patients (IoT devices) and MCs may coexist. The patients can be everywhere in the area.

For a simple representation of the system, a few assumptions were made. First, all MCs within the grid are considered candidates for edge-server placement. Second, all MCs are virtually connected to each other if and only if $d(MCi, MCj) = K$, in which, $d(a, b)$ is a distance function for calculating the distance between two points (a and b), and K is the largest possible distance to connect the neighbours (MCs). The last assumption is that all MCs have different volume and distribution of user requests (historical loads) over a long period.

The set of MCs (candidate points) is, hence, defined as $MCs = \{mc_1, mc_2, ...; mc_n\}$, where each refers to a preselected, feasible placement location in the grid (in coordinate axes) and $noMCs =$ number of MCs in that area. A set of edge servers is denoted by $ES(es1, es2, ...; esk)$ and $Nes =$ number of edge-servers needed to be placed for full coverage. It is assumed that all edge servers are identical (homogeneous). A set of patients requiring edge computing services is denoted by $P = \{p_1, p_2, ...; p_m\}$, where $m =$ number of patients. A cost function $Cost(es_j, mc_k)$ is defined to refer to the incurred cost of placing an edge server $es \in ES$ at an MC $mc_k \in MCs$.

Two latency functions, L_{ij} and D, are defined to represent the latency when patient p_i is served by edge servers placed at mc_j of the grid and the latency between two edge servers, respectively. It is assumed the bandwidth across the grid is homogeneous; hence, latency is predominantly affected by distance. The main objectives are to reduce the cost of deploying edge-servers in the region and reduce the latency in accessing edge services with full coverage to all patients. The edge-servers placement problem (ESPP) is now formulated as a multi-objective optimisation model. The following decision variables are defined:

- X represents the placement of edge-servers at MCs

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

Where:

$$x_j = \begin{cases} 1, & \text{if } es_i \text{ placed at } mc_j \\ 0, & \text{otherwise.} \end{cases}$$

- Y represents the assignment of patents to MCs

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

Where:

$$y_{ij} = \begin{cases} 1, & \text{if } p_i \text{ placed at } mc_j \\ 0, & \text{otherwise.} \end{cases}$$

- E represents the links between MCs

$$E = \{e_{ab} \mid 1 \leq a, b \leq n, a \neq b\}$$

where:

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

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

$$\lambda_{sum} = \sum_{i=1}^{m} \lambda_i \tag{4.1}$$

The SRT is calculated based on the previous study:

$$SRT = \frac{\sum_{i=1}^{n} t_{pi}}{n} \tag{4.2}$$

Where:

$$\begin{aligned} t_{pi} = T_{wi} + E_{AP} * T_{AP} + (c_1 + 1) * t_{MC} + c_1 * V + c_2 * S \\ + c_2 * t_H + c_3 * L + c_3 * t_H + c_4 * B + c_4 * t_{Cloud} \end{aligned} \tag{4.3}$$

The edge-servers placement problem (ESPP) is defined as follows. Given a system model parameter $(G, noMCs\ [points], K, Nes, m, L, D, \lambda_{MAX})$, the problem is to find X (the placement of edge-servers) among the MCs, such that the system response time SRT is minimised, i.e.,

$$Min \sum Cost(es_j, mc_k).X_j \tag{4.4}$$

$$Min\ \ SRT \tag{4.5}$$

$$Min \sum E \tag{4.6}$$

Subject to:

1. $N_{es} \geq 3$: to ensure that at least three edge-servers are placed at three MCs to preserve previous achievements in chapter 3.

2. $d(mc_a, mc_b) = k$: to ensure a shorter distance between a patient and an MC and to avoid the colocated problem.

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

4. $\sum y_{ij} = 1 \;\; (1 \leq j \leq n)$: to guarantee that all patients must be served, and each is served from exactly one candidate point.

4.4 Optimal edge-servers placement (OESP)

The main idea is how to determine the best MC locations for deploying edge servers in order to provide services to all patients in a given area. The number of selected MCs depends on the total number of MCs in a certain area, that ensure that services are provided to all patients, and how they are distributed and connected to each other. This algorithm is built based on several stages, as follows.

Step 1: Locate all coordinates on the map.

- This step involves identifying and recording the coordinates of all relevant points on the map.
- Ensure that the map is valid and that there are no repeated points.

In this step, the algorithm focuses on locating and recording the coordinates of all relevant points on the map. It is important to ensure that the map is valid, meaning it accurately represents the desired area or region, and that there are no repeated points. The uniqueness of points is crucial to prevent any duplication or confusion during subsequent stages of the algorithm.

Step 2: Generate important matrices:

1. Distance Matrix (*DistancesBtAll*): Generate a matrix that represents the distances between all pairs of points. This matrix helps quantify the proximity between points.

2. Connectivity Matrix (*ConnectivityMatrix*): Generate a matrix that indicates the connectivity between points, where 1 represents a connection and 0 represents no connection.

After that, establish virtual connections between neighbouring points based on a specified maximum distance threshold ($Max\_DistanceToConnect$).

Step 3: Finding and selecting the first, second and third best points.

In this step, the algorithm evaluates points based on multiple criteria, including the number of links per point (*SumLinks*), the sum of distances to other points (*SumDistances*), and historical loads *Load_History*. The objective is to select the best points that ensure connectivity to the highest possible number of points.

The algorithm generates a matrix of points sorted based on these three criteria. The primary criterion is the number of links per point (*SumLinks*), which prioritises points that can establish connections with the greatest number of other points. In cases where two or more points have the same number of connections, the algorithm applies the second criterion, which is the sum of distances to other points (*SumDistances*). This criterion focuses on selecting points that are closest to other points in terms of distance. If a tie persists even after considering the second criterion, the algorithm resorts to the third criterion, which is historical loads *Load_History*.

By sorting the points in the matrix according to these criteria, the algorithm aims to identify and select the first, second, and third best points. To ensure optimal connectivity, it is important to consider the following conditions for the second and third best points:

1. Direct Connection to the First Best Point: The second and third best points must be directly connected to the first best point, establishing a direct link between them.

2. No Connection between Second and Third Best Points: The second and third best points should not be connected to each other. This arrangement ensures that the second and third best points are chosen on both sides of the first best point, maximising the number of points connected on both sides.

By adhering to these conditions, the algorithm guarantees that the selected points promote effective connectivity and facilitate the connection of as many points as possible to the three best points in the network.

Step 4: Test Connectivity 1

In this step, the algorithm performs a connectivity test to evaluate the network's current state and identify connected and disconnected points. The goal is to determine if all points are successfully connected or if there are any points that remain unconnected. The algorithm carries out the following actions:

1. Determine the Connected Points.

2. Determine the NotConnected Points.

By analysing the connectivity status of the points based on the previous selections, the algorithm can assess the network's current state. If all points are connected, indicating that every point is successfully linked to the network, the algorithm terminates as it has achieved its goal. However, if there are still unconnected points, the algorithm proceeds to the next stages to address and resolve the connectivity issues.

Step 5: Identify Points Unreachable for Connection (Distant Points) and Update the Unconnected Ones.

In this step, the algorithm focuses on identifying points that are too far away to establish connections with other points in the network. These distant points are unlikely to be reachable and cannot be effectively integrated into the current network configuration. The algorithm subsequently updates the list of unconnected points based on this assessment.

This step is significant as it helps determine which points are geographically distant and cannot be connected. By isolating these distant points, it becomes possible to consider alternative strategies, such as integrating them with other working regions or adjusting the network configuration to accommodate their unique circumstances.

Step 6: Determine the *Other_Best* points from the Connected points.

In this step, the algorithm identifies additional best points from the pool of connected points, one by one, to further enhance the network's connectivity. After each selection, the algorithm updates the list of unconnected points based on the newly established connections. The number of these additional best points is not fixed, but rather, a sufficient number should be chosen to achieve full network connectivity while ensuring that the previously mentioned conditions and constraints are met.

Step 7: Test Connectivity 2

In this step, the algorithm performs a second connectivity test to evaluate the network's current state and ensure that all points, except those previously determined as unreachable, are connected. The goal is to verify that the number of disconnected points is zero, indicating successful network connectivity. If there are still disconnected points remaining, the algorithm repeats Step 6 to further enhance the connectivity. Algorithm 2, the OESP algorithm, represents the proposed algorithm for optimising the placement of intelligent edge-servers in the healthcare field.

Algorithm 2: Optimal Edge-Servers placement (OESP) algorithm

Input: $noMCs$, $X = \{x_i, 1 \leq i \leq noMCs\}$, $Y = \{y_i, 1 \leq i \leq noMCs\}$, $Max\_DistanceToConnect$, $Load\_History$

Output: Find the best points (Best_Points)

Constraints: $Best\_Points \geq 3$, $no.hopstotheBests = 1$

Step 1: Determine the coordinates

Step 2: Generate important matrices:

```
$index \leftarrow 1$
for $k \leftarrow 1$ to $noMCs$
    for $m \leftarrow 1$ to $noMCs$
        $DistancesBtAll(index) \leftarrow point_k – point_m$
        if $DistancesBtAll(index2) \leq DistanceToConnect$ then:
            $ConnectivityMatrix(k, m) \leftarrow 1$
        else
            $ConnectivityMatrix(k, m) \leftarrow 0$
        end if statement
        index++
    end for statement
end for statement
```

Step 3: Find Best three points

```
1- **Create:** $Point\_Features \leftarrow [SumLinks\ \ SumDistances\ \ Load\_History]$
2- **Sort:** $Point\_Features \leftarrow [SumLinks(desc)\ \ SumDistances(asc)\ \ Load\_History(desc)]$
Then **Find:**
a- $First\_Best \leftarrow$ First member of $Point\_Features$(sorted).
b- Second_Best
for i ← 2 to noMCs then
    if $Point\_Features$ (i,1) is connected to $First\_Best\_Point$
        $Second\_Best \leftarrow Point\_Features$ (i,1)
        break for loop
    end if statement
end for statement
c- $Third\_Best$
if $Point_Features(i,1)$ in connected to $First\_Best\_Point$ ...
AND not connected to $Second\_Best\_Point$
    $Third\_Best \leftarrow Point\_Features(i,1)$
    break for loop
else
    $Third\_Best \leftarrow Point\_Features(i,1)$
end if statement
```

Step 4: Test Connectivity 1

```
$i \leftarrow 1$ , $k \leftarrow 1$
for $j = 1$ to $noMCs$
    if $j \neq First\_Best$ AND $j \neq Second\_Best$ AND $j \neq Third\_Best$
        if $ConnectivityMatrix(j, First\_Best) \neq 1$ AND
        $ConnectivityMatrix(j, Second\_Best) \neq 1$
        AND $ConnectivityMatrix(j, Third\_Best) \neq 1$
            $NotConnected(i) \leftarrow j$
            $i++$
        else
            $Connected(k) \leftarrow i$
           $K++$
        end if statement
    end if statement
end for statement
```

Step 5: Find Never connected Points (Distant points) (NeverConnected)

```
$k \leftarrow 1$
for $i \leftarrow 1$ to $no\_Notconnected$
```

```
    for $j \leftarrow 1$ to $no\_connected$
        if $ConnectivityMatrix(i, j) == 0$
            $NeverConnected(k) \leftarrow i$
            $NotConnected(i) \leftarrow [\,]$
           $K++$
       end for statement
    end for statement
end for statement
```

Step 6: Find other Best point (if needed)

```
while 1
    if $no\_NotConnected \geq 1$
        $k = 1$
        for $s \leftarrow 1$ to $no\_Connected$
            if $s$ is connected to $NotConnected$
                $Current\_Best(k) \leftarrow s$ (To find the candidate points to be among the best.)
                $k++$
            end if statement
        end for statement
        k = 1
        for $i \leftarrow 1$ to $no\_NotConnected$
            for $j \leftarrow 1$ to $no\_Current_Best$
                if $ConnectivityMatrix(i, j) == 1$
                    Choose a one with a shorter distance.
                    $Other\_Best(k) \leftarrow j$ (with a shorter distance)
                   $k++$
                end if statement
            end for statement
        end for statement
    else
```

```
        break while loop
    end if statement
end while statement
```

Stage 7: Test Connectivity 2

4.5 Results

In this section, data is randomly generated to evaluate the performance of the proposed algorithms. Table 4.1 summarises the simulation parameters used to calculate the System Response Time (SRT), which represents the average system latency required to deliver patient data to one of the health facilities within the HMAN architecture. Additional parameters will be introduced subsequently.

4.5.1 Priority Mechanism based on AI

The initial step in processing the collected data involves wearable devices worn by patients to filter the data and transmit any abnormal data to the local MC for further processing. Upon reaching the local MC, priority is given to patients with the most severe conditions. In our simulation, three scenarios were conducted to evaluate the performance of the proposed priority mechanism. To assess the effectiveness of the algorithm, a comparison was made between the SRT for accessing the service with and without the application of the algorithm.

The first scenario assumed that 100 patients (data or workload) arrived at the local MC at the same time. The first step is to check whether the local MC is capable of handling the received data or not. If it can, the data is processed at the local MC and the services are delivered to the patients without needing to send out the data to other units. However, if it is determined that the MC cannot handle the data of all 100 patients, then it must offload a part of it to other units based on the HOSSC

Table 4.1: SRT System parameters

Symbol	**Parameter**	**Value**
Ps	Number of patients	100, 200, 300
n	Number of servers in each MC (GP)	5, 6, 7
m	Number of servers in each GH	10, 12, 14
r_{Wi}	Link rate between IoT device and AP	54 Mbps
r_{AP}	link rate between AP and the local GP	100 Mbps
r_{Fib}	link rate between the GPs and GHs	Up to 10 Gbps
λ_i	Packet size	30 KB
T_{wi}	Wireless Delay	4 ms
T_{AP}	Delay between AP and the local GP	2 ms
V	Delay between two neighboring GPs	$\frac{(1-K_1).\lambda_{GPsum}}{10Gbps}$
S	Delay between the GPs and GHs	$\frac{K_2.\lambda_{GHsum}}{10Gbps}$
L	Delay between two neighboring GHs	$\frac{(1-K_2).\lambda_{GHsum}}{10Gbps}$
B	Internet Delay	20 ms
μ_{GP}, μ_{GH}	Each GP, GH server service rate	100,200 KB per ms
μ_{Cloud}	Cloud service rate	1000 KB per ms
λ_{GPmax}	Maximum GP workload	200×n KB
λ_{GHmax}	Maximum GH workload	200×n KB

algorithms. The main task of the proposed priority mechanism is to determine which patients are served by the local MC and which ones should be offloaded. In this scenario, the algorithm classified the received data into four classes and then, served the patients accordingly. Patients with the highest priority and greatest need are served in a shorter time compared to those with less severe conditions. The same approach is applied in the 200 and 300 patient scenarios, where the algorithm categorises the data and prioritises the patients accordingly.

It is worth noting that the absence of a priority mechanism, the timing of service provision to patients would be uncertain since the HMAN architecture does not distinguish between patients. Table 4.2 shows the results obtained from applying the above scenarios and demonstrates the effectiveness of the proposed algorithm in providing services with less delay to the patients with the highest level of urgency. In the case of HMAN, it is not possible to determine which patient should be treated first, because the system handles data equally without taking urgency into account. This could potentially have negative consequences on the treatment of patients with severe conditions.

Table 4.2: SRT with each class with/without applying the proposed priority mechanism.

No. patients	**Classes**		**SRT (n = 5, m = 10)**	
	Type	**No. patients**	**No Priority Mechanism (HMAN)**	**With Priority Mechanism**
100	A	3	**N/A**	Discarded
	B	61		18-19 ms
	C	28		8-9 ms ms
	D	8		5-6 ms ms
200	A	5	**N/A**	Discarded
	B	103		25 ms
	C	63		16-17 ms ms
	D	29		7-8 ms ms
300	A	4	**N/A**	Discarded
	B	181		32 ms
	C	73		19-20 ms ms
	D	42		9-10 ms ms

4.5.2 Optimal edge-servers placement (OESP) algorithm

In this section, several simulations are conducted to evaluate the performance of the proposed algorithms. The algorithm was tested on randomly generated points, with each point representing an MC site. Table 4.3 lists the parameters defined for the OESP algorithm. To demonstrate the algorithm's effectiveness, the process began with 10 randomly generated points, detailing each step to identify the optimal points for placing edge servers. Subsequently, the final results obtained from applying the algorithm to 20 and 30 randomly generated points are presented.

Table 4.3: OESP simulation parameters.

Symbol	**Definition**	**value**
A	The region dimensions	100 km $\times$ 100 km
noMCs	Number of Points (MCs)	10, 20, 30
x, y	Coordinates	Randomly created
D_{max}	Max. distance between two neighbours	40 km

- ***Scenario 1 (10 nodes)***

Firstly, a MATLAB code generated 10 points in a geographic area of 100 km $\times$ 100 km. Every point was considered as an MC site, as mentioned before. Then, these points were assigned to their respective locations on a virtual map, as depicted in Figure 4.2.

The subsequent task involved establishing virtual connections between these points based on the first constraint, which specified a maximum distance of 40 km to connect two points. As depicted in Figure 4.3, there were multiple redundant links between certain points. The objective of the proposed algorithm was to minimise the number of these connections while ensuring that the entire network remained connected.

The next step involved determining the best three points based on the number of links per point, proximity to other points, and the historical loads per point, which

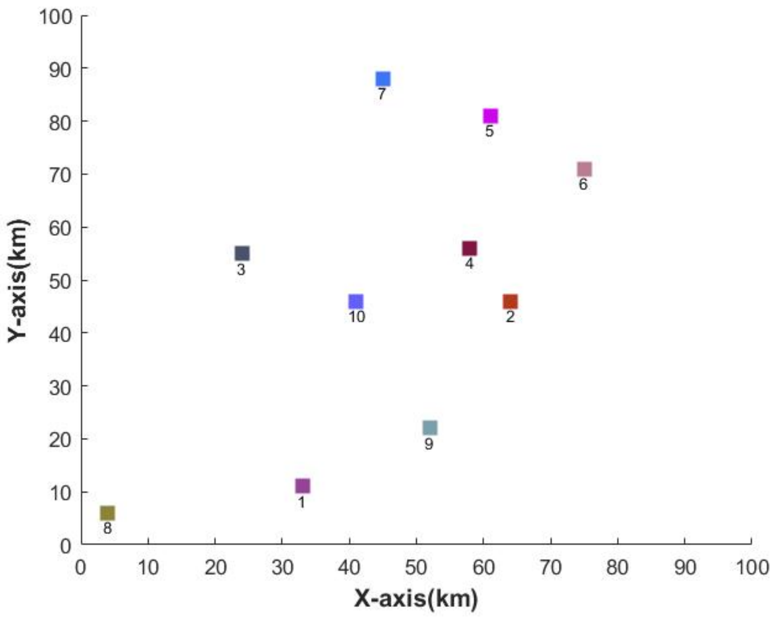


Figure 4.2: 10 disconnected points (MCs) on a map.

were assumed randomly in this simulation. Referring to Table 4.4, which ranks the point according to these three parameters, it is clear that the point 4 is the first-best one, because it has more links than the others. The second-best point is 10, as it is directly connected to point 4 and has more links compared to the remaining points. In order to enhance network coverage, the algorithm disregarded point 2 (or point 9) as the third-best point, despite its higher rank compared to the others. This decision was based on the fact that point 2 (or point 9) shares a direct link with point 10, causing them to fall on the same side of the best point. Instead, point 5 was chosen as the third-best point, fulfilling the specified conditions and situated on a distinct side from the best point. After choosing these three best points, the algorithm establishes connections between all the other points and its closest best point while removing all other links. Figure 4 depicts the network after applying the previous steps. Note that the point 8 remains unconnected as it is located more than 40 km away from all the best points. Therefore, the next algorithm task will be to identify other best points to ensure complete connectivity.

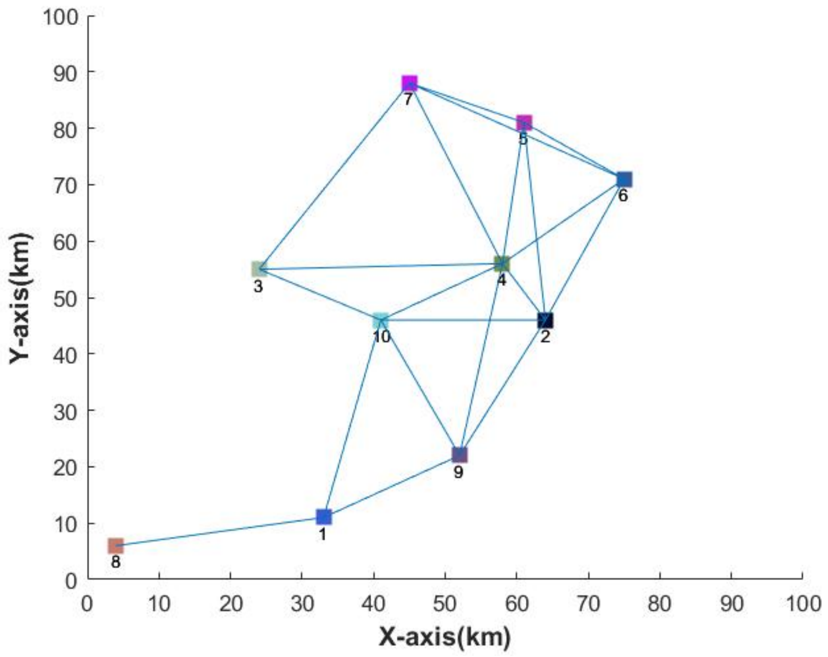


Figure 4.3: A virtual network of 10 connected points (MCs) based on distances between neighbours (Dmax = 40 km).

Before moving on to the next step, it is crucial to highlight that the algorithm generated four matrices that depict the current state of the points within the network. These matrices are as follows:

- Best = [4 10 5]
- Connected = [2 9 6 7 3 1]
- NotConnected = [8]

The subsequent step involves determining which points should be selected to be the best ones. The algorithm does not impose any limitation on the number of these points. The primary objective is to achieve full network connectivity while minimising costs and adhering to all constraints. The remaining unconnected points need to be checked for the possibility of connecting them or not. If the distance between an unconnected point and any connected points is less or equal to the maximum distance (constrain 1), then this point can be connected. Subsequently, the algorithm selects the closest Connected point to it and includes this Connected point among the Other-

Table 4.4: Sorted 10 points (MCs) based on the number of links per point, how close the point is to all others, and the historical loads per point.

Points	**Number of links**	**Distance to all**	**Historical loads**
4	7	307.3874	711
10	5	303.4536	211
2	5	329.8995	459
9	4	383.7702	253
5	4	409.7863	602
6	4	420.9528	718
7	4	449.8674	847
3	3	373.1516	393
1	3	456.99	424
8	1	615.2826	879

Best points. Conversely, if this condition is not met, it is isolated from the group and marked as a never-connected point. Figure 4.5 illustrates that point 8 is connected to point 1 which is selected as an Other-Best point.

After completing this step, the matrices have been updated as follows:

- Best = [4 10 5]
- Other_Best = [1]
- Connected = [2 9 6 7 3]
- NotConnected = ∅
- Never_Connected = ∅

With the current configuration, all points are connected, and no points remain unconnected. This indicates that complete coverage has been achieved in the network, fulfilling the objective of the algorithm.

Upon observing Figure 4.5, it becomes evident that there is a noticeable difference between the network configuration before and after the algorithm was applied. The

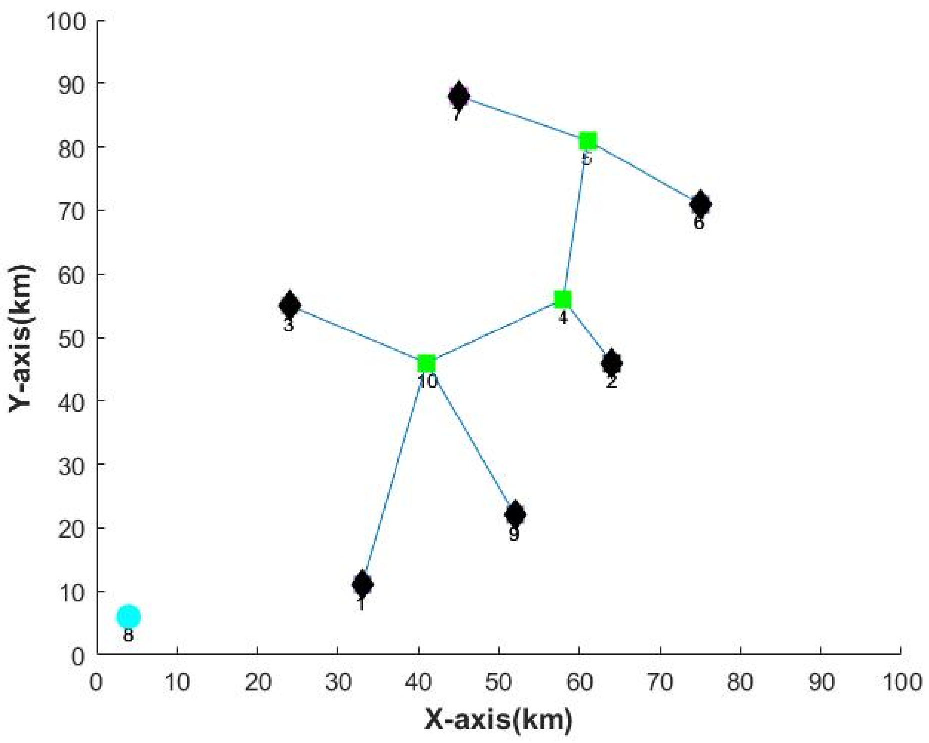


Figure 4.4: A network of 10 points after choosing the first, second, and third best points (MCs): Green points = Best points, Black points = connected points and Cyan = Not-connected points.

algorithm was able to successfully select 4 points out of 10 to achieve complete network connectivity. Each point is now within one hop distance from the nearest Best point and with not more the maximum distance constraint. Through this process, several goals were achieved, including reducing cost and delay. Instead of deploying servers at all ten points, they are now strategically placed at only four points, leading to a significant reduction in the number of connections required. Additionally, the algorithm has successfully reduced latency to the greatest extent possible by considering the distance and number of hops required to reach the servers. By strategically placing the servers at selected points, the algorithm ensures that data transmission distances are minimised, resulting in reduced latency. This optimisation of network infrastructure contributes to improved overall performance and responsiveness of the system.

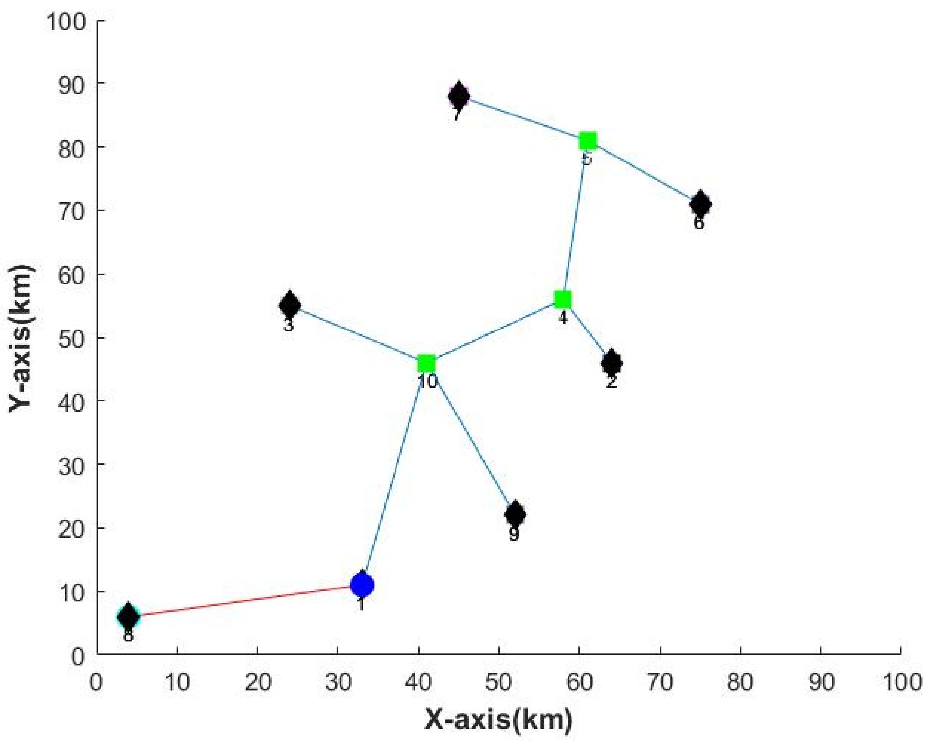
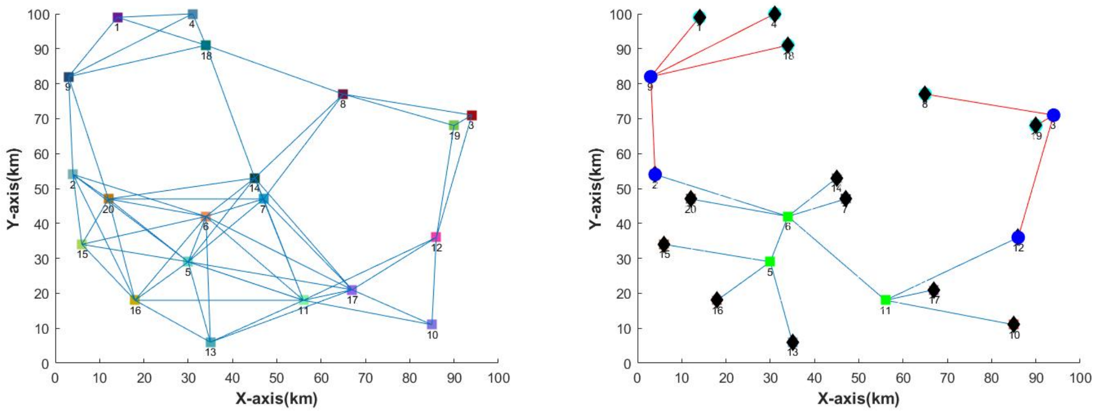


Figure 4.5: The final network of 10 points: Green points = Best points, Blue pints = Other best points, Black points = connected points.

- ***Scenarios 2 and 3 (20 and 30 points respectively)***

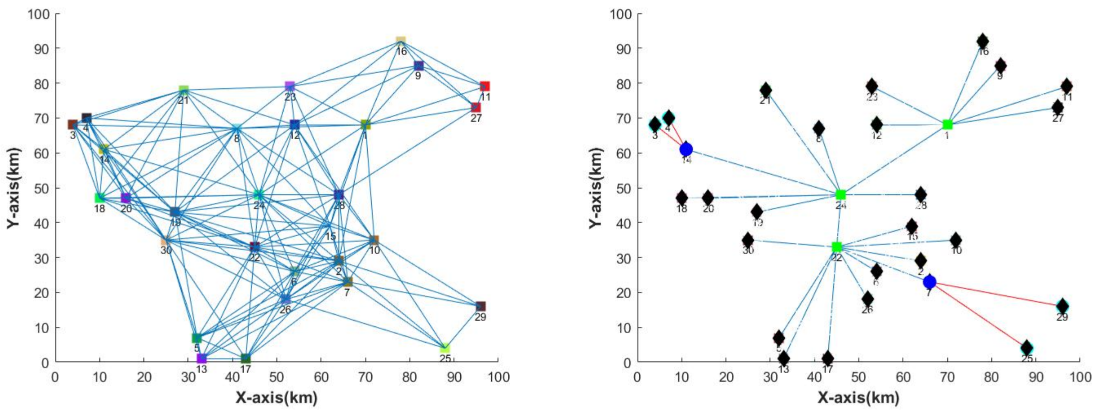
To demonstrate the effectiveness of the proposed algorithm, Figures 4.6 and 4.7 show two different scenarios involving the selection of the Best points to position edge-servers among a pool of 20 and 30 points, respectively. Figures 4.6b and 4.7a present the final shape of the networks of 20 and 30 points after applying the algorithm. The initial steps of the algorithm involve the identification of the Best three points and assessing the network's connectivity. Subsequently, the algorithm proceeds to select additional Other-Best points to attain complete network connectivity, while adhering to predefined constraints.

The crucial point that needs to be emphasised, and can be clearly seen when examining the results, is that the selection of the best points and determination of their number depends on the shape of the network and the proximity of the points



(a) (b)

Figure 4.6: A network of 20 points before (a) and after (b) choosing the best points (MCs): Green points = Best points, Blue pints = Other best points, Black points = connected points.



(a) (b)

Figure 4.7: A network of 30 points before (a) and after (b) choosing the best points (MCs): Green points = Best points, Blue pints = Other best points, Black points = connected points.

to each other. There is no fixed method or specific number of points that must be chosen in all cases. For instance, in the 20-point networks, seven points were selected as the best points, while the 30-point network only required five points to be chosen as the best for full connectivity (see Figures 4.6 and 4.7). Furthermore, this algorithm ensures minimal delay in accessing services. Users can directly connect to an MC with edge-servers or through an MC that is one hop away from the edge-servers. By applying this algorithm, several benefits have been achieved, including a significant reduction in cost while maintaining low latency. The proposed algorithm ensures that the delay, which was a key achievement of the HMAN system, is preserved. Figure 4.8 illustrates the comparison of latency before and after implementing the algorithm, demonstrating that the algorithm successfully maintains the low delay achieved by the HMAN system. Minimising latency enhances the network's overall performance and efficiency, directly benefiting patient care. In medical settings, timely data processing and rapid response times are critical, where delays can significantly impact patient outcomes. This preservation of low delay ensures efficient access to services for users.

Table 4.5 provides a comparison that highlights the cost difference‘ between the HMAN system before and after the application of the OESP algorithm. It is worth noting that these percentages may vary across different networks, as they are influenced by the number of points selected as the Best points. The percentages are determined by calculating the difference in the number of edge servers before and after the application of OESP. This difference is then divided by the number of edge servers present before OESP. For instance, in the case of 30 points, the number of edge servers before OESP is 30, which is subsequently reduced to a range between 5 and 12 by OESP, depending on the shape of the network and the proximity of the points to each other. Therefore, the calculations are as follows: (12 - 30) / 30 = -60% to (5 - 30) / 30 = -83%. However, in all cases, there is a significant improvement in cost while preserving the previously achieved benefits outlined in chapter 3.

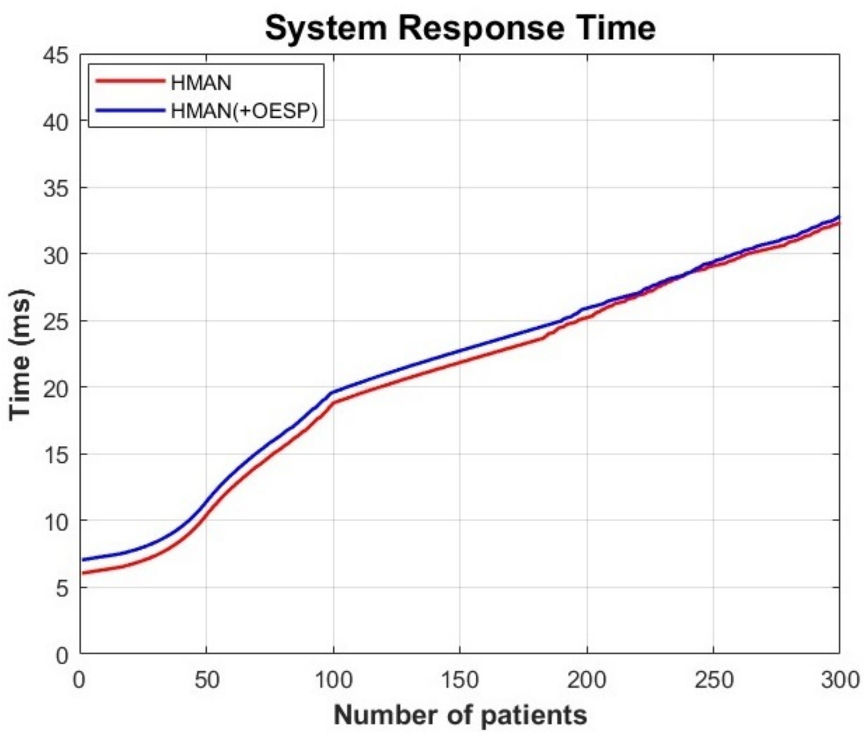


Figure 4.8: A comparison of latency before and after implementing OESP algorithm on HMAN architecture.

Table 4.5: A comparison between HMAN and OESP cost

noMCs	**No. of edge-server nodes in HMAN**	**No. of edge-server nodes in OESP**	**Deployment Cost**
3	3	3	0
5	5	3	-40%
10	10	3-5	-(50-70)%
15	15	5-7	-(53-66)%
20	20	5-9	-(55-75)%
30	30	5-12	-(60-83)%

In short, this study introduces a novel algorithm for edge server placement in health monitoring frameworks. By considering the shape of networks and proximity of nodes, the OESP algorithm overcomes limitations observed in existing literature reviews. Unlike previous approaches, such as [103], [104] and [105], that rely on network size for edge server selection, it offers a more robust and tailored solution. In addition, one significant advantage of this study is the careful selection of sites, ensuring that they are located within one hop from the connected sites. This strategic placement of edge servers minimises latency and contributes to the overall effectiveness of the proposed algorithm. By reducing the distance and number of hops required to reach the servers, the latency is kept to a minimum, resulting in improved performance and a seamless user experience. The findings of this research contribute to advancing the field of edge computing in healthcare systems, opening avenues for further exploration and optimisation in this domain.

4.6 Summary

The primary motivation of s-health is to contribute to reducing hospitalisation rates, while providing affordable telehealth services to remote patients. By integrating s-health with edge/fog computing, additional benefits, such as reduced delay and power consumption, network bandwidth savings, as well as improved security and data privacy can be achieved. However, a key challenge lies in determining the optimal placement of edge-servers in a cost-effective manner while ensuring full coverage for all patients with minimal latency. In this study, two algorithms have been proposed with the aim of providing an efficient priority offloading/processing mechanism and solving the edge-server placement problem. The simulation results have shown that the two proposed algorithms are highly promising. The priority mechanism algorithm successfully classified patients based on the severity of their disease and prioritised their services accordingly. On the other hand, the Optimal Edge-Server Placement (OESP) algorithm effectively identified optimal locations for deploying edge-servers, achieving objectives such as cost reduction with minimal delay. Although the proposed algorithms showcased promising results in improving the efficiency and effectiveness of edge-server placement and priority offloading/processing, further research is needed to address areas such as load balancing and resource allocation for fully optimising network performance. In summary, the combination of s-health, edge/fog computing, and the proposed algorithms offers a comprehensive solution for delivering cost-effective and efficient telehealth services. This research opens up new avenues for improving healthcare accessibility, reducing costs, and enhancing patient care through advanced technologies and intelligent algorithms.

Chapter 5

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

5.1 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 [137].

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 [138]. This approach has several key benefits, including increased system efficiency, faster performance, and lower latency. By reducing the load on each server, load balancing minimises the risk of network failures and improves the overall responsiveness of applications by distributing the workload evenly. 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. 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 [139, 140].

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 [141, 142, 143].

SDN-based load balancing algorithms are a type of dynamic load balancing algorithm that utilise Software-Defined Networking (SDN) capabilities to manage and distribute network traffic. These algorithms provide centralised 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 utilising SDN technology, SDN-based load balancing algorithms can offer a more dynamic and effective way to balance network traffic and optimise network performance [144, 145, 146].

This chapter 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 study are as follows:

1. A load balancing framework is introduced for healthcare systems in urban areas, leveraging edge computing and software-defined networking (SDN) technology.

2. The Load Balancing of Optimal Edge-servers Placement (LB-OESP) algorithm is proposed, an enhanced version of the OESP algorithm [147] proposed in Chapter 4. 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, surpassing the approach in Chapter 4. Additionally, 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 favourable outcomes, including a cost-effective system and reduced latency compared to previous work.

Chapter Organisation: The structure of this chapter is organised as follows. Section 5.2 presents the proposed system model and problem formulation. In Section 5.3, the load balancing algorithms are introduced. The performance evaluation of the proposed algorithms is discussed in Section 5.4, where the results are presented and analysed . Finally, Section 5.5 provides a summary of the study.

5.2 Preliminaries

5.2.1 Proposed System Model

Figure 5.1 presents a conceptual framework for healthcare systems in urban areas that leverages edge computing and software-defined networking (SDN) technology. This model embodies a modern approach to healthcare, utilising edge computing and SDN technology to deliver fast, efficient, and reliable healthcare services to patients. The framework comprises various components, including Medical Centres (MCs), Hospitals (Hs), and Cloud, in conjunction with patients located throughout the entire region.

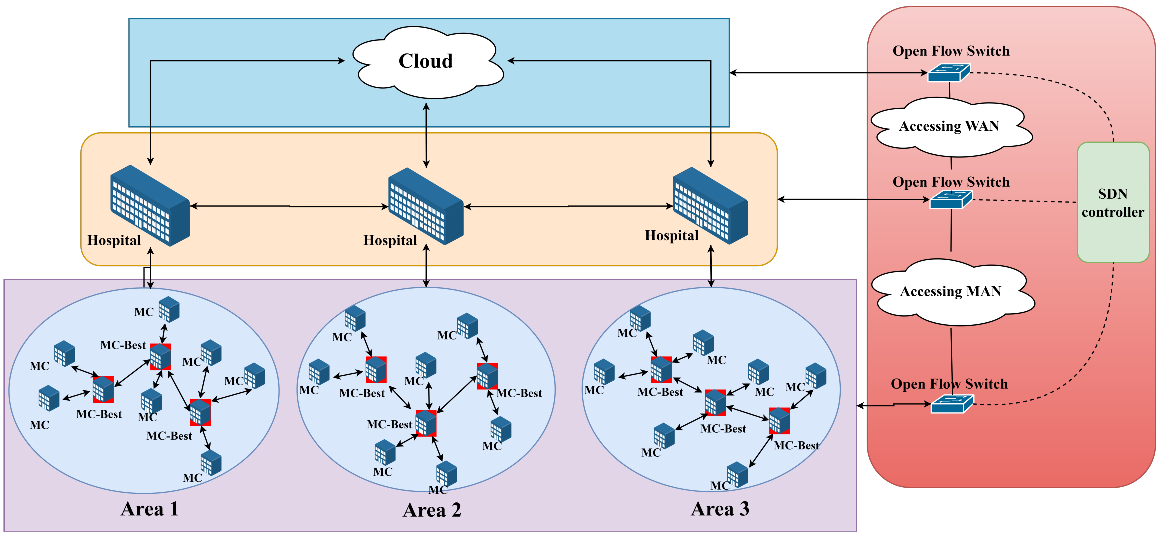


Figure 5.1: Proposed Edge/Fog healthcare architecture based on SDN technology.

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

The urban environment is conceptualised as a three-dimensional grid that comprises of three distinct entities, namely, Hospitals (Hs), Medical Centers (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 have been established. Firstly, it is assumed that every MC situated within the grid can potentially serve as a suitable candidate for the placement of edge-servers. Secondly, all MCs are considered to be virtually connected 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 utilised for establishing connectivity between neighboring MCs. Lastly, it is assumed 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 can be expressed as $MCs = \{mc_1, mc_2, ...; mc_n\}$, where each mc denotes a feasible placement location within the three-dimensional grid, and $noMCs$ represents the number of MCs within that specific region. The set of edge servers required for complete coverage in this area is represented as $ES = \{es_1, es_2, ...; es_k\}$, with Nes denoting the total number of edge servers necessary. It is important to note that all edge servers are considered identical (i.e., homogeneous). The patient population is denoted by $P = \{p_1, p_2, ...; p_m\}$, where m corresponds to the total number of patients in the region.

5.2.2 Problem Formulation

According to Chapter 4, 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 edge servers, with the constraint that at least three MCs must be designated as such to enable efficient data offloading to neighbouring MCs. Conversely, MCs-Others are designated as serverless nodes that must be directly connected to MCs-Best to prevent multi-hop transfers and minimise 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 Chapter 3. 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 utilisation of resources and minimising 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. Firstly, the utilisation of Software-Defined Networking (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. Secondly, the proposed approach involves improving the OESP algorithm for selecting the best server sites and reducing them if possible. This approach aims to minimise the number of MCs designated as MCs-Best, while still ensuring full coverage for all MCs-Others. By optimising 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 multi-objective optimisation model has been developed, with the following decision variables defined:

- X represents the placement of edge-servers at MCs

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

Where:

$$x_j = \begin{cases} 1, & \text{if } es_i \text{ placed at } mc_j \\ 0, & \text{otherwise.} \end{cases}$$

- Y represents the assignment of patents to MCs

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

Where:

$$y_{ij} = \begin{cases} 1, & \text{if } p_i \text{ placed at } mc_j \\ 0, & \text{otherwise.} \end{cases}$$

- E represents the links between MCs

$$E = \{e_{ab} \mid 1 \leq a, b \leq n, a \neq b\}$$

where:

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

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

$$\lambda_{sum} = \sum_{i=1}^{m} \lambda_i \tag{5.1}$$

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

$$SRT = \frac{\sum_{i=1}^{n} t_{pi}}{n} \tag{5.2}$$

Where:

$$\begin{aligned} t_{pi} = {} & T_{wi} + E_{AP} * T_{AP} + (c_1 + 1) * t_{MC} + c_1 * V + c_2 * S \\ & + c_2 * t_H + c_3 * L + c_3 * t_H + c_4 * B + c_4 * t_{Cloud} \end{aligned} \tag{5.3}$$

$$t_{MC} = f_Q(K_1.\lambda_{MCsum}) + \frac{K_1.\lambda_{MCsum}}{\mu_{MC}} \tag{5.4}$$

$$t_H = f_Q(K_2.\lambda_{H_sum}) + \frac{K_2.\lambda_{Hsum}}{\mu_H} \tag{5.5}$$

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 Chapters 3 and 4 for all tasks generated by patients within the network while considering the delay requirements of task execution in healthcare systems. The optimisation problem can be formulated as follows: given a system model with parameters $(G, noMCs\ [points], K, Nes, m, T_{wi}, T_{AP}, D, \lambda_{MAX})$, the goal is to find X among the best MCs and balance the system to enhance the response time SRT.

$$Min \sum Cost(es_j, mc_k).X_j \tag{5.6}$$

$$Min\ \ SRT \tag{5.7}$$

Subject to:

1. $N_{es} \geq 3:$ to ensure that at least three edge-servers are placed at three MCs to preserve previous achievements in chapter 3.
2. $d(mc_a, mc_b) = k:$ to ensure a shorter distance between a patient and an MC and to avoid the colocated problem.
3. $\sum E_{ij} = 1 \quad or \quad 2:$ to ensure that a patient either connects directly with an edge-servers site or through only one MC to minimise the latency.
4. $\sum y_{ij} = 1 \quad (1 \leq j \leq n):$ to guarantee that all patients must be served, and each is served from exactly one candidate point.

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

5.3 Proposed Load Balancing algorithms

The key objective of the previous chapter was to effectively deploy edge-servers to MCs within a region to suit the demands of all monitored patients. The study utilised the OESP algorithm, which demonstrated success in identifying the optimal placement of edge-servers 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.

The primary aim of this study 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 summarised as follows:

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

Step 2: Implement a static load balancing algorithm to reorganise 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 summarised 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 need 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.
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.

```
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 and SRT
Step 1: Define the maximum coverage
- Combine all MCs-Best in a one matrix called $MCs\_Best$.
- Combine all MCs-Other in a one matrix called $MCs\_Others$.
for each $MCs\_Best$, do
    Plot a circle with a radius of K = 40 km.
end
Step 2: Find Best-MCs-Groups matrix
- Greate an $MCs\_Best\_details$ matrix = 0.
for each $MCs\_Best$, do
    Find $MC\_Best\_Group$ matrix.
    $MCs\_Best\_details = MCs\_Best\_details + MC\_Best\_Group$.
end
Step 3: Find Best of Best MCs
for each $MCs\_Best$, do
    if $MC\_Best_i\_Group \subseteq MC\_Best_j\_Group$
        Delete $MC\_Best_i$ from $MCs\_Best$ matrix
        Add $MC\_Best_i$ to $New\_Other$ matrix
    end
end
$Best\_MCs\_Best$ = updated $MCs\_Best$
```

```
$MCs\_Others = MCs\_Others + New\_Other$
```

Step 4: Test Connectivity (1)

```
Generate $ConnectivityBest = 0$
for $i \leftarrow 1$ to size-of($Best\_MCs\_Best$), do
    for $k \leftarrow 1$ to size-of($Best\_MCs\_Best$), do
        if $i \neq k$, then
            if distance between $Best\_MCs\_Best_i$ and $Best\_MCs\_Best_j \leq 40$, 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
```

Step 5: Choose a New MC-Best

```
for each $Not\_Connected\_MCs$, do
    if distance between $Not\_Connected\_MCs$ and $New\_Other \leq 40$, then
        Add $New\_Other$ to $Best\_MCs\_Best$
    end
end
```

Step 6: Test Connectivity (2)

Step 7: Re-plot the MCs position in the map

Step 8: Find new connections between the $Best\_MCs\_Best$

```
for $i \leftarrow 1$ to size-of($Best\_MCs\_Best$), do
    for $k \leftarrow 1$ to size-of($Best\_MCs\_Best$), do
        if $i \neq k$, then
            Find the nearest $Best\_MCs\_Best$
        end
    end
    Connect the nearest $Best\_MCs\_Best$ to $Best\_MCs\_Best_i$
- Plot the new links between $Best\_MCs\_Best$
```

Step 9: Find the number of MCs_Others **that should connected to each MC-Best with load balancing.**

```
$No\_Other\_BestMC = \frac{No_Other}{No_Best}$
$R1 = int(No\_Other\_BestMC)$        (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 $Best\_MCs\_Best$ to achieve best load
  balancing
```

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

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 software-defined networking (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 utilisation of available resources significantly, resulting in enhanced processing capabilities and optimal network utilisation.

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, a binary decision variable $T_{x1}^{x_2}$ can be defined as:

$$T_{x1}^{x_2} = \begin{cases} 1, & \text{if } t_{offloading} < t_{local} \\ 0, & \text{otherwise.} \end{cases} \tag{5.8}$$

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 equips the controller to make more informed predictions, including average propagation delays, queuing 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 optimise SRT and load balancing, Algorithm 2 proposes an SDN-based greedy heuristic algorithm (SDN-GH). 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 Equation 5.8.

Algorithm 2: SDN-based Greedy Heuristic (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

Step 1: The respective SDN controller receives periodic state reports from each MC and H.

```
Step 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
```

```
Step 3
for each $\lambda_{sum}$ arrived at the same time, the SDN do
    if $\lambda_{sum} < C_{MC}$ then
        Execute $\lambda_{sum}$ 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
```

```
Step 4
for each $\lambda_{off}$ arrived at the same time, the SDN do
    Calculate $t_{offloading}$.
    Calculate $t_{local}$.
end
```

```
Step 5
According to Equation 5.8:
set the value of $T_x^{x_i}$ between the local MCs ($x$) and MCs neighbour $xi = \{x1, x2\}$
and the value of $T_x^H$ between the local MCs ($x$) and the local H.
if $T_x^{x_i}$ and $T_x^H$ == zero then
    Execute $\lambda_{off}$ at the local MC.
    Record the response time.
    Terminate the process.
else
    Construct $\beta_1 = \{T_x^{x_1}, T_x^{x_2}, T_{x_1}^H\}$.
end

Step 6
If $\beta_1 \neq \emptyset$ then
    Determine the MC or H index $k = min\beta_1$.
    Offload $\lambda_{off}$ to $k$ for the execution.
    The steps from 1-6 are applied to H with its neighbours and the cloud.
else
    Execute $\lambda_{off}$ at the local MC.
    Record the response time.
    Terminate the process.
end

Step 7 Return: the total response time.
```

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 neighbouring MCs-Best according to their available capacities (**Scenarios 2 and 3**). If the neighbouring 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 neighbouring Hs (**Scenarios 5 and 6**). If the neighbouring 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**). Figure 5.2 visualises the seven scenarios. It is worth noting that in the previous study in Chapter 3, the implementation of the HOSSC Algorithm led to the successful achievement of five data processing scenarios, characterised by enhanced privacy, elevated computational capability, minimised system latency, sustained high service availability, and a scalable system design. In this study,the number of data processing scenarios was expanded to seven, which enabled us to improve upon the previously established outcomes while ensuring load balance and reducing delay time.

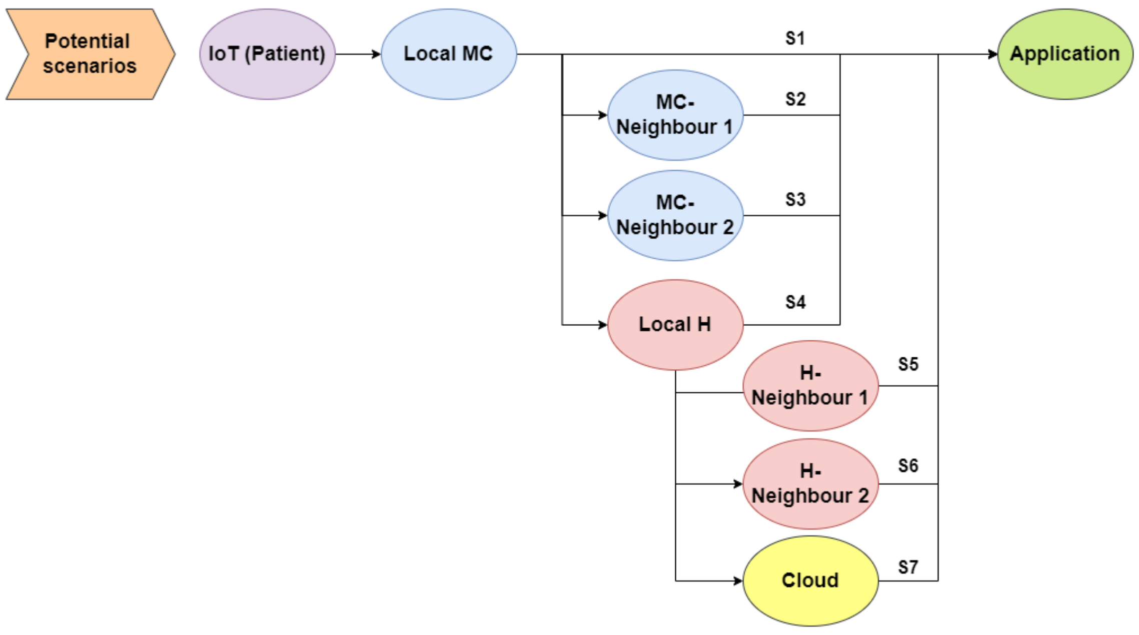


Figure 5.2: All potential scenarios based on SDN-GH.

5.4 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(R) Core(TM) i7-8550U CPU, 16 GB RAM, Microsoft Windows 10 operating system, and MATLAB R2022b. Table 5.1 lists the simulation parameters, primarily derived from our previous studies in Chapter 3 and 4. Additionally, 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 utilisation 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.

5.4.1 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 (Chapter 4) serves as the input for the LB-OESP algorithm, identical parameters are defined for

Table 5.1: SRT System parameters

Symbol	**Parameter**	**Value**
Ps	Number of patients	100, 200, 300
n	Number of servers in each MC (GP)	5, 6, 7
m	Number of servers in each GH	10, 12, 14
r_{Wi}	Link rate between IoT device and AP	54 Mbps
r_{AP}	link rate between AP and the local GP	100 Mbps
r_{Fib}	link rate between the GPs and GHs	Up to 10 Gbps
λ_i	Packet size	30 KB
T_{wi}	Wireless Delay	4 ms
T_{AP}	Delay between AP and the local GP	2 ms
V	Delay between two neighboring GPs	$\frac{(1-K_1).\lambda_{GPsum}}{10Gbps}$
S	Delay between the GPs and GHs	$\frac{K_2.\lambda_{GHsum}}{10Gbps}$
L	Delay between two neighboring GHs	$\frac{(1-K_2).\lambda_{GHsum}}{10Gbps}$
B	Internet Delay	20 ms
μ_{GP}, μ_{GH}	Each GP, GH server service rate	100,200 KB per ms
μ_{Cloud}	Cloud service rate	1000 KB per ms
λ_{GPmax}	Maximum GP workload	200×n KB
λ_{GHmax}	Maximum GH workload	200×n KB

both algorithms, as detailed in Table 5.2. To demonstrate the effectiveness of the approach, the step-by-step process of the algorithm is showcased initially with 10 randomly generated, followed by the presentation of final outcomes when applying the algorithm to 20 and 30 randomly generated points.

- ***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

Table 5.2: LB-OESP simulation parameters.

Symbol	**Definition**	**value**
A	The region dimensions	100 km $\times$ 100 km
noMCs	Number of Points (MCs)	12, 16, 25
x, y	Coordinates	Randomly created
D_{max}	Max. distance between two neighbours	40 km

points on a virtual map, as depicted in Figure 5.3.

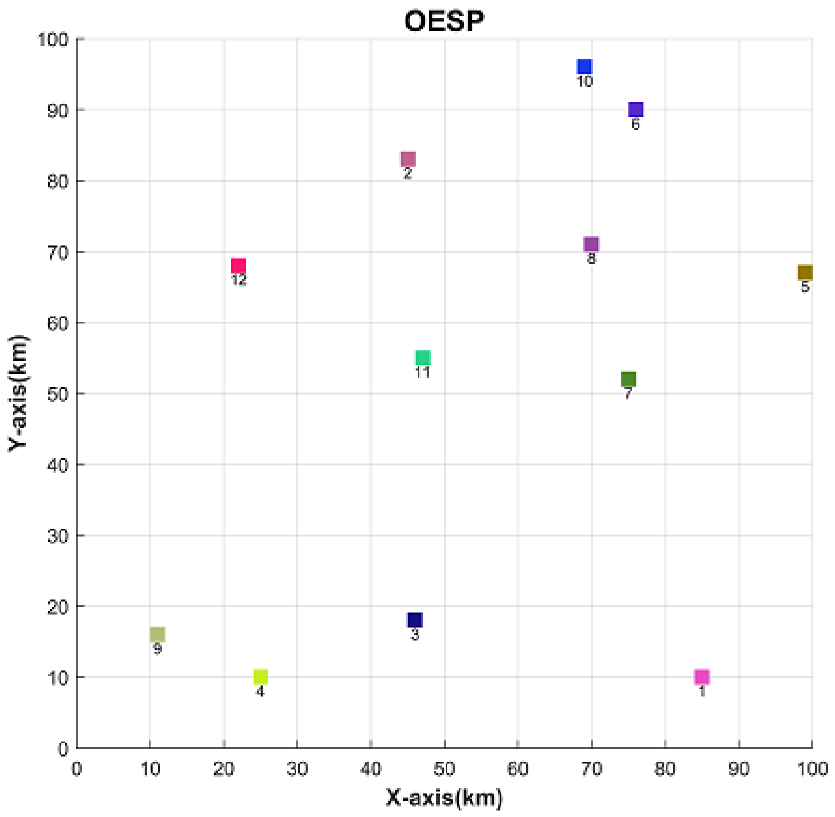


Figure 5.3: 10 disconnected MCs on a map.

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 Figure 5.4.

The OESP algorithm demonstrates success in selecting the optimal MCs-Best,

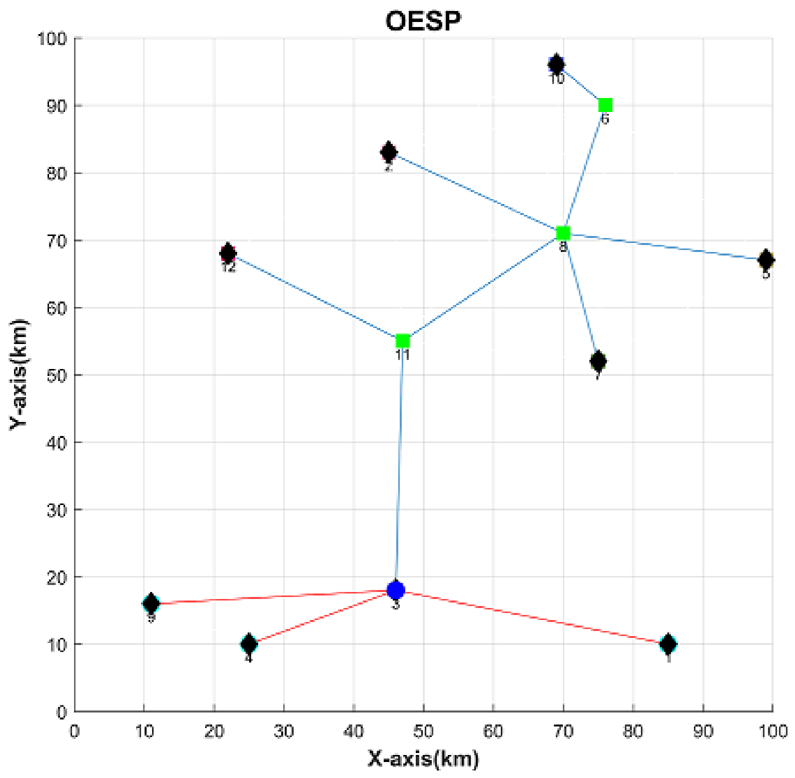


Figure 5.4: The final OESP network of 12 MCs: Green & Blue points = Best-MCs, Black points = MCs-Others.

which are determined to be (6, 8, 11, and 3). The algorithm successfully establishes a connection between each MC-Others and a single MC-Best. Despite this achievement, it is evident that the distribution of loads within the network generated by this algorithm is not uniform. Specifically, MCs-Best (11 and 6) are each connected to only one MC-Others, whereas MCs-Best 8 and 3 are connected to three MC-Others each, thus leading to an imbalance in the load distribution.

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 minimise 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 to 6 of the algorithm. Once completed, the network will be reconfigured, with connections between sites being adjusted, as illustrated in Figure 5.5.

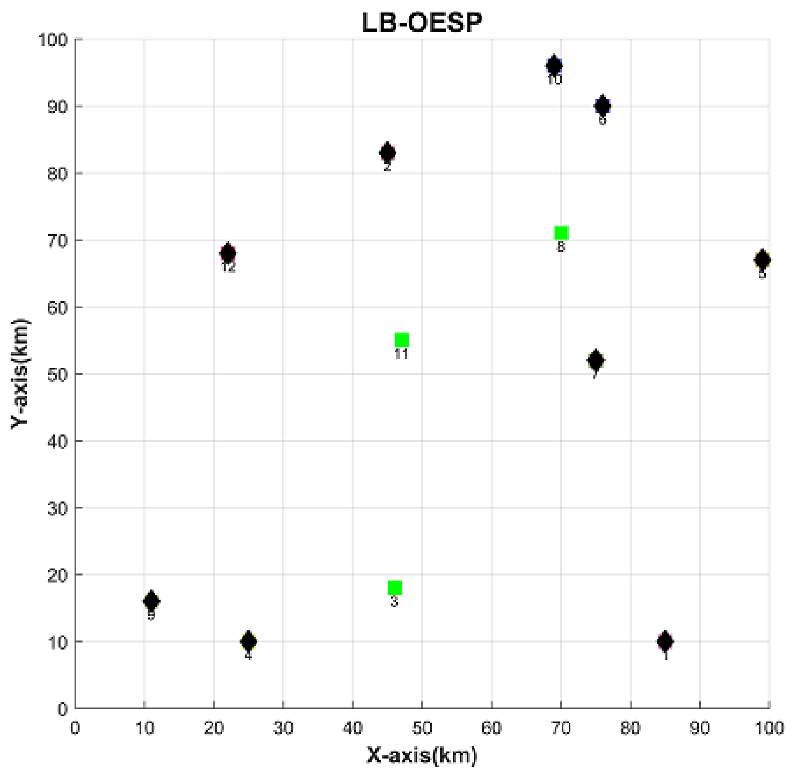


Figure 5.5: 12 disconnected MCs on a map. Green points = Best-MCs-Best, Black points = MCs-Others.

As depicted in Figure 5.5, 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. Figure 5.6 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 num-

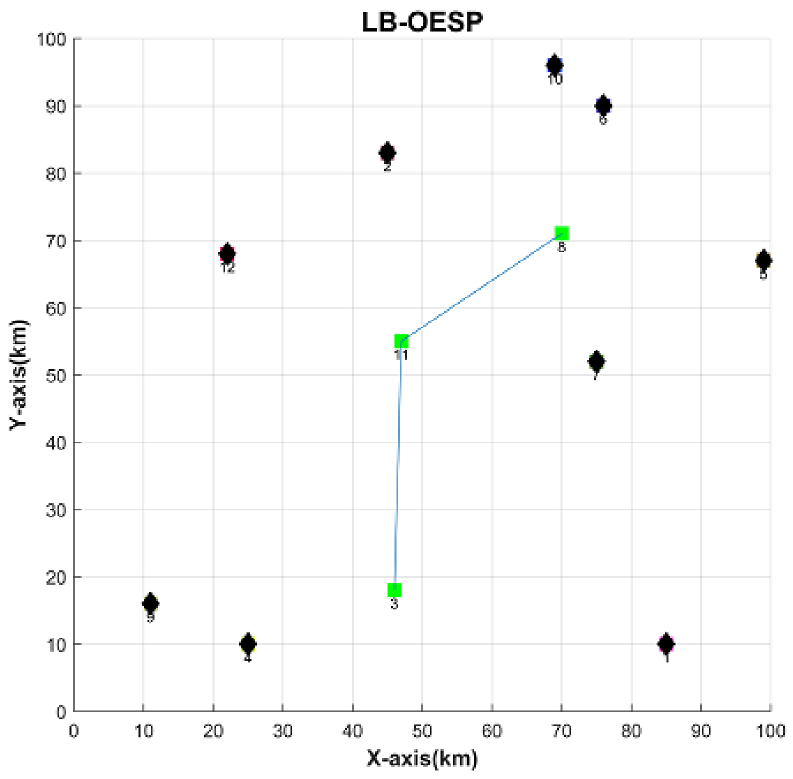


Figure 5.6: The established connections between the best sites in the network.

ber of MC-Others to be connected to each MC-Best, considering the minimum and maximum threshold. In this particular scenario, with 9 MC-Others and 3 MCs-Best, the optimal number was 3 MC-Others (both as the minimum and maximum) to be connected to each MC-Best, taking into account the proximity of the interconnected sites. Figure 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.

As illustrated in Figure 5.7, 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 exam-

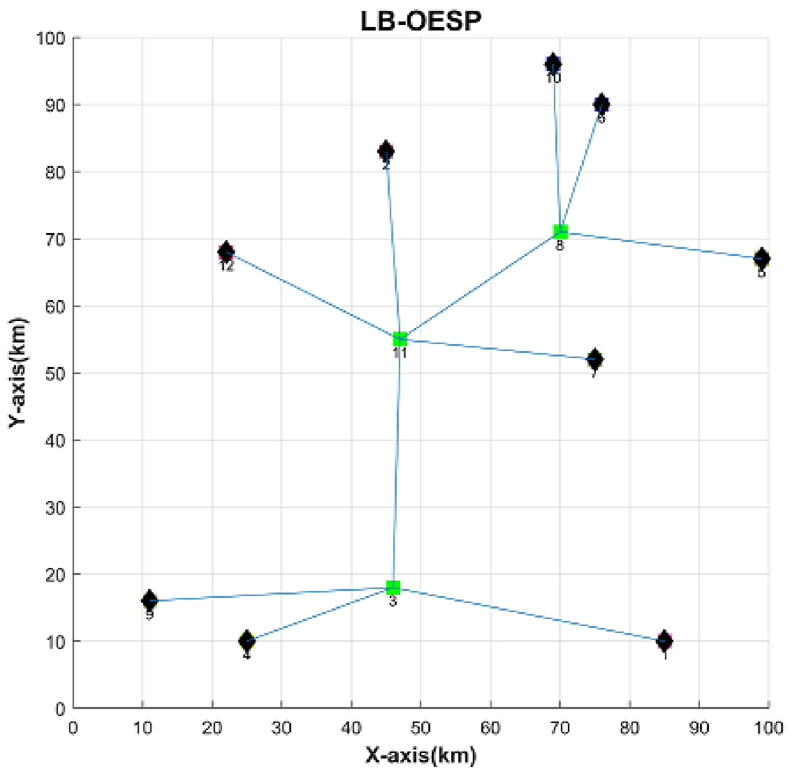


Figure 5.7: The final LB-OESP network of 10 MCs: Green points = Best-MCs-Best, Black points = MCs-Others.

ple, 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 the case of disconnection, the algorithm proceeds by adding one MC at a time from the isolated matrix until full connectivity between them is achieved.

- ***Scenario (2 and 3) – 16 and 25 nodes***

The effectiveness of the proposed algorithm is further highlighted in Figures 8 and 9, which illustrate two distinct scenarios involving 16 and 25 nodes, respectively. These figures showcase the output maps generated by both the OESP algorithm and the subsequent LB-OESP algorithm. The algorithm consistently demonstrates its ability to minimise 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.

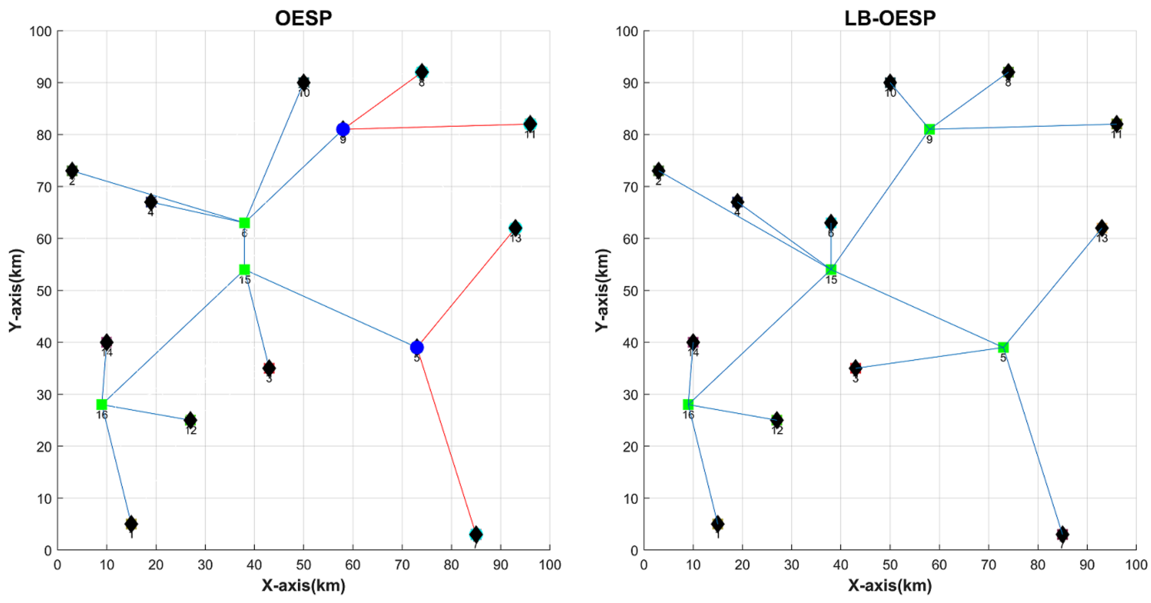


Figure 5.8: A comparison between the final OESP and LB-OESP network of 16 MCs.

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.

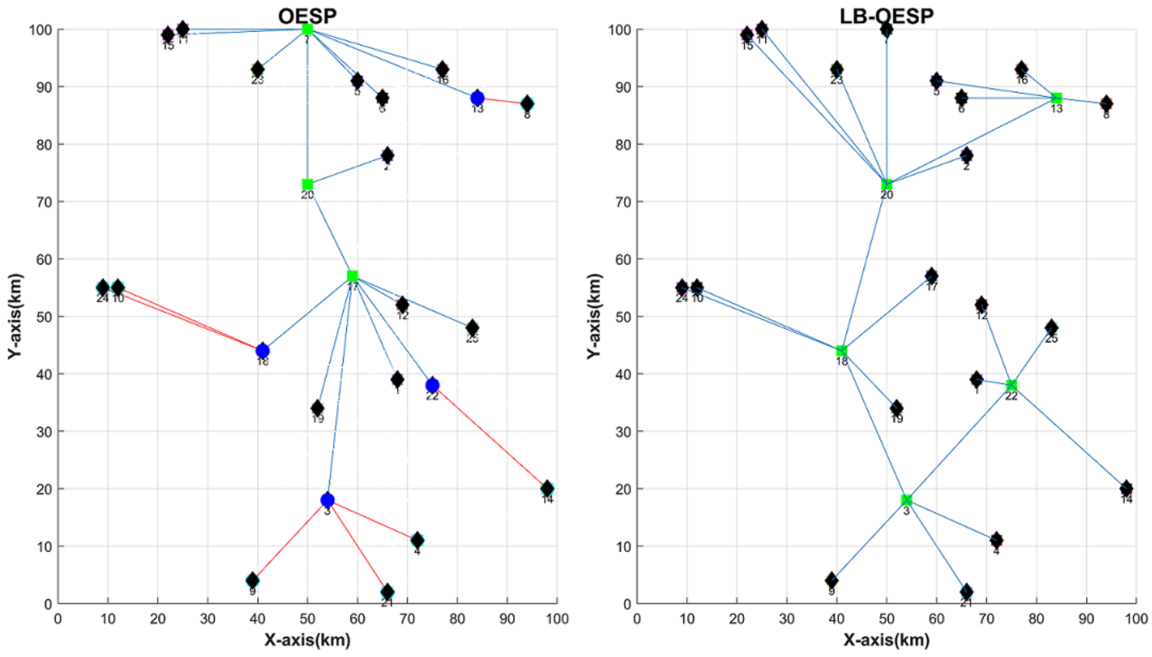


Figure 5.9: A comparison between the final OESP and LB-OESP network of 25 MCs.

Table 5.3 provides a cost comparison between the proposed algorithm and previous studies in Chapters 3 and 4. It is worth noting that these percentages may vary across different networks, as they are influenced by the number of points selected as Best points. The percentages are determined by calculating the difference in the number of edge servers between LB-OESP and OESP. This difference is then divided by the number of edge servers in OESP. For instance, in the case of 25 points, the number of edge servers in OESP is 7, which is subsequently reduced to 5 by LB-OESP, depending on the shape of the network and the proximity of the points to each other. Therefore, the calculations are as follows: (5 - 7) / 7 = -28%. However, in all cases, there is a significant improvement in cost while preserving the previously achieved benefits outlined in chapters 3 and 4.

Table 5.3: A comparison between HMAN, OESP and LB-OESP cost

noMCs	No. of edge-server nodes in			Deployment Cost
	HMAN	OESP	LB-OESP	
12	12	4	3	-25%
16	16	5	4	-20%
25	25	7	5	-28%

5.4.2 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 Figure 5.10, 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, are employed. To facilitate a more straightforward evaluation of the system's viability and benefits, several parameters have been defined and presented in Table 5.1, which are identical to the parameters used in previous studies.

It is evident that the algorithm endeavors to minimise 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

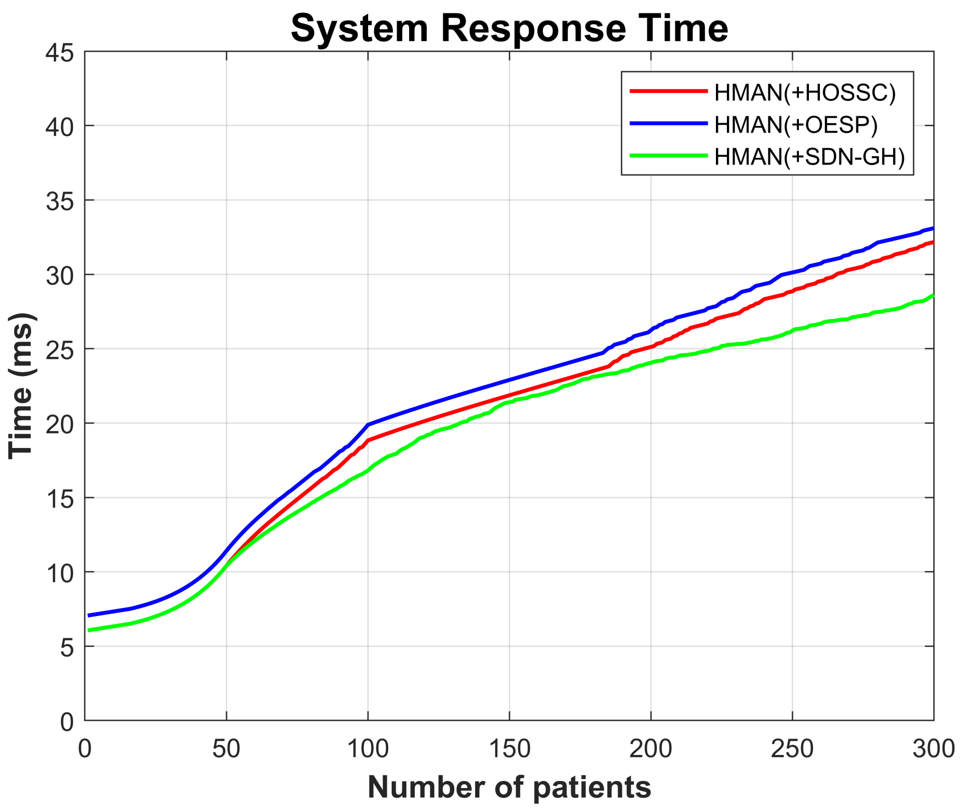


Figure 5.10: A comparison between HOSSC, OESP and SDN-GH algorithms in terms of the SRT in the system.

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 centres (M_{CN1} and M_{CN2}). 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 Figure 5.2.

Furthermore, the results indicate a significant enhancement in the system's performance when utilising the SDN-GH algorithm compared to the HOSSC and OESP algorithms, with SDN-GH succeeding in reducing the delay by approximately 12% (28ms - 32ms / 32ms = 12%). 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. Additionally, this reinforces the privacy and availability of the system while contributing to scalability and capacity improvements, as previously reported in Chapter 3.

5.5 Summary

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 edge server 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.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This research undertook the conceptualization and exploration of a cooperative hierarchical healthcare architecture, designated as HMAN, amalgamating the potential of Edge/Fog computing. The architectural proposal, enriched by the support of the HOSSC algorithm, has engendered a framework comprising four distinctive layers: IoT, Edge/Fog, cloud, and application layers. Through meticulous model analysis and extensive experimentation, this research has ascertained the manifold benefits that the HMAN architecture entails.

Notably, the architecture's adaptability to varied network workloads has been successfully demonstrated, reaffirming its resilience and robustness in a dynamic operational context. Additionally, the study's focus on ensuring data privacy and low latency through locally processed data scenarios has underscored the architecture's practicality. The introduction of a comprehensive five-stage evaluation framework has facilitated the architecture's meticulous assessment, further corroborating its performance and viability.

The empirical results underscore the architecture's prowess, revealing commendably low latencies ranging from 6.043 to 31.45 milliseconds across diverse workloads. These outcomes reflect the architecture's inherent computational capacity, scalability, and availability, affirming its potential to enhance healthcare service delivery.

The integration of s-health with Edge/Fog computing has been a critical focal point of this study, aimed at revolutionizing remote telehealth services. The fusion of these domains presents an array of benefits, encompassing reduced latency, lower power consumption, enhanced data privacy, and improved security. However, the endeavor also entails intricate challenges, including optimal edge-server placement and efficient workload prioritization.

The study's contributions in the form of two novel algorithms have been instrumental in addressing these challenges. The proposed priority offloading/processing mechanism and the Optimal Edge-Server Placement (OESP) algorithm have showcased significant promise. The priority mechanism algorithm adeptly classifies patients based on disease severity and allocates resources accordingly, while the OESP algorithm effectively determines optimal edge-server locations for cost-efficient deployment with minimal latency.

The third study's exploration centered on a pivotal challenge in smart healthcare systems—efficient load balancing. This critical facet significantly influences system performance, avoiding overburdening nodes, delays, and compromised efficiency. The integration of static and SDN-based load balancing algorithms has yielded a comprehensive approach to attaining optimal load distribution in Edge/Fog-based healthcare networks [148].

The results showcased impressive outcomes, including a 12% reduction in system latency and up to 28% decrease in deployment costs compared to previous studies. The efficacy of the proposed approach has been further highlighted by the SDN-GH algorithm's dynamic load balancing capabilities, leading to heightened system performance. Moreover, the implementation of the SDN-GH algorithm has facilitated seven distinct data processing scenarios, elevating privacy, computational capabilities, system latency, service availability, and system adaptability.

6.2 Future Directions and Enhancements

The research presented in this thesis offers a robust foundation for advancing the realm of healthcare architectures and computing paradigms. The insights gained from this work lay the groundwork for numerous promising future directions and enhancements, each aimed at further augmenting the efficacy, scalability, and real-world applicability of the proposed systems.

1. **Cross-Country Network Expansion:** The current research has primarily focused on local healthcare networks within cities. An exciting avenue for future exploration involves expanding the scale of these networks to span across countries (WAN). Investigating the dynamics of cross-country healthcare networks introduces unique challenges and opportunities, including latency management, data privacy regulations, and diverse healthcare infrastructure. Such an expansion can usher in new dimensions of healthcare accessibility and collaboration on a global scale.

2. **Facility Status Monitoring Protocol:** To streamline communication processes and enhance response times, the development of a specialized protocol tailored for Software-Defined Networking (SDN) environments could be pursued. This protocol would facilitate real-time monitoring of facility statuses within the network. By continuously assessing the operational conditions of network facilities, this protocol would enable proactive adjustments, minimizing downtime, optimizing resource utilization, and consequently elevating the overall responsiveness of the system.

3. **Advanced Resource Allocation Algorithms:** Resource allocation stands as a pivotal concern in healthcare networks. Future research can delve into designing resource allocation algorithms that dynamically adapt to changing demands and network conditions. These algorithms could consider variables such as patient priority, workload diversity, and resource availability to optimize resource distribution and system efficiency.

4. **Intelligent User-Edge Server Assignment:** The allocation of users to edge servers within specific areas warrants comprehensive exploration. The design of a sophisticated method for user-edge server assignment can mitigate potential issues that might impact service quality and availability. This method could incorporate factors like user load, proximity, and edge server capabilities to ensure optimal user experiences while maximizing resource utilization.

5. **Strengthening Security Measures:** Security remains a paramount concern for healthcare systems. Future work can delve deeper into security enhancements by exploring advanced cryptographic techniques and secure multi-party computation. These techniques can fortify data protection mechanisms, ensuring the confidentiality and integrity of sensitive patient information amidst a dynamic and interconnected network environment.

6. **Leveraging Blockchain for Data Management:** The integration of blockchain technology holds the potential to revolutionize patient data management and transactional integrity. By harnessing the immutable and transparent nature of blockchain, the system's security and trustworthiness can be elevated. Research in this direction could focus on developing a blockchain-based framework to ensure secure and auditable patient data exchange, thereby enhancing the overall reliability of the ecosystem.

7. **Real-World Deployment and Validation:** To bring the theoretical advancements into practical fruition, real-world deployment within healthcare environments is essential. Collaboration with healthcare institutions to implement and evaluate the systems in clinical settings will provide invaluable insights. This practical feedback loop will enable refinement of the proposed systems based on real-world challenges and requirements, further solidifying their effectiveness and applicability.

The envisioned future directions outlined above elucidate a spectrum of opportunities for building upon the research conducted in this thesis. By delving into these areas, the proposed systems can be refined, extended, and validated, ultimately contributing to the advancement of healthcare technology and service delivery.

Finally, It is crucial to highlight that while the proposed framework and algorithms are specifically designed for healthcare applications, their adaptability extends beyond this domain. With some modifications, these solutions can be effectively applied to various other fields such as the Industrial Internet of Things (IIoT) and autonomous vehicles. The underlying achievements and principles of the framework are versatile, making it suitable for these applications as well. This flexibility underscores the robustness and broad potential impact of the developed technologies, paving the way for their utilization in diverse and innovative ways across multiple industries.

Bibliography

[1] S. Oueida, Y. Kotb, M. Aloqaily, Y. Jararweh, and T. Baker, “An edge computing based smart healthcare framework for resource management,” *Sensors*, vol. 18, no. 12, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/12/4307

[2] J. M. Corchado and S. Trabelsi, “Advances in sustainable smart cities and territories,” p. 1280, 2022.

[3] K. Amirthalingam, “Medical dispute resolution, patient safety and the doctor-patient relationship,” *Singapore medical journal*, vol. 58, no. 12, p. 681, 2017.

[4] I. Balansard, L. Cleverley, K. L. Cutler, M. G. Spångberg, K. Thibault-Duprey, and J. A. Langermans, “Revised recommendations for health monitoring of non-human primate colonies (2018): Felasa working group report,” *Laboratory animals*, vol. 53, no. 5, pp. 429–446, 2019.

[5] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, and S. K. Das, “Edge-computing-driven internet of things: A survey,” *ACM Comput. Surv.*, vol. 55, no. 8, dec 2022. [Online]. Available: https://doi.org/10.1145/3555308

[6] Y. Zhao, W. Wang, Y. Li, C. Colman Meixner, M. Tornatore, and J. Zhang, “Edge computing and networking: A survey on infrastructures and applications,” *IEEE Access*, vol. 7, pp. 101 213–101 230, 2019.

[7] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, “A survey on the edge computing for the internet of things,” *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[8] P. Mach and Z. Becvar, “Mobile edge computing: A survey on architecture and computation offloading,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[9] H. J. Damsgaard, A. Ometov, and J. Nurmi, “Approximation opportunities in edge computing hardware: A&nbsp;systematic literature review,” *ACM Comput. Surv.*, vol. 55, no. 12, mar 2023. [Online]. Available: https://doi.org/10.1145/3572772

[10] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762118306349

[11] W. G. Hatcher, J. Booz, J. McGiff, C. Lu, and W. Yu, “Edge computing based machine learning mobile malware detection,” 2017.

[12] W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, and M. Satyanarayanan, “Quantifying the impact of edge computing on mobile applications,” in *Proceedings of the 7th ACM SIGOPS Asia-Pacific workshop on systems*, 2016, pp. 1–8.

[13] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, “Lavea: Latency-aware video analytics on edge computing platform,” in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–13.

[14] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, “Edge computing: A survey,” *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18319903

[15] S. Hamdan, M. Ayyash, and S. Almajali, “Edge-computing architectures for internet of things applications: A survey,” *Sensors*, vol. 20, no. 22, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/22/6441

[16] S. Raza, S. Wang, M. Ahmed, M. R. Anwar *et al.*, “A survey on vehicular edge computing: architecture, applications, technical issues, and future directions,” *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[17] S. B. Calo, M. Touna, D. C. Verma, and A. Cullen, “Edge computing architecture for applying ai to iot,” in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 3012–3016.

[18] C. M. Fernández, M. D. Rodríguez, and B. R. Muñoz, "An edge computing architecture in the internet of things," in *2018 IEEE 21st international symposium on real-time distributed computing (ISORC).* IEEE, 2018, pp. 99–102.

[19] M. Marjanović, A. Antonić, and I. P. Žarko, "Edge computing architecture for mobile crowdsensing," *IEEE Access*, vol. 6, pp. 10 662–10 674, 2018.

[20] B. Ali, M. A. Gregory, and S. Li, "Multi-access edge computing architecture, data security and privacy: A review," *IEEE Access*, vol. 9, pp. 18 706–18 721, 2021.

[21] O. Abari, D. Bharadia, A. Duffield, and D. Katabi, "Enabling {high-quality} untethered virtual reality," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 2017, pp. 531–544.

[22] W. Yu, H. Xu, H. Zhang, D. Griffith, and N. Golmie, "Ultra-dense networks: Survey of state of the art and future directions," in *2016 25th international conference on computer communication and networks (ICCCN).* IEEE, 2016, pp. 1–10.

[23] J. Gedeon, J. Krisztinkovics, C. Meurisch, M. Stein, L. Wang, and M. Mühlhäuser, "A multi-cloudlet infrastructure for future smart cities: An empirical study," in *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, 2018, pp. 19–24.

[24] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE communications surveys & tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[25] P. Bellavista and A. Zanni, "Feasibility of fog computing deployment based on docker containerization over raspberrypi," in *Proceedings of the 18th international conference on distributed computing and networking*, 2017, pp. 1–10.

[26] Y. Mansouri and M. A. Babar, "A review of edge computing: Features and resource virtualization," *Journal of Parallel and Distributed Computing*, vol. 150, pp. 155–183, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520304317

[27] C. Martín Fernández, M. Díaz Rodríguez, and B. Rubio Muñoz, "An edge computing architecture in the internet of things," in *2018 IEEE 21st International Symposium on Real-Time Distributed Computing (ISORC)*, 2018, pp. 99–102.

[28] G. Wang, Y. Zhao, J. Huang, and W. Wang, "The controller placement problem in software defined networking: A survey," *IEEE network*, vol. 31, no. 5, pp. 21–27, 2017.

[29] Y. Jararweh, A. Doulat, A. Darabseh, M. Alsmirat, M. Al-Ayyoub, and E. Benkhelifa, "Sdmec: Software defined system for mobile edge computing," in *2016 IEEE international conference on cloud engineering workshop (IC2EW)*. IEEE, 2016, pp. 88–93.

[30] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, oct 2019. [Online]. Available: https://doi.org/10.1145/3362031

[31] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *2017 Global Internet of Things Summit (GIoTS)*, 2017, pp. 1–6.

[32] Y. Pan, P. Thulasiraman, and Y. Wang, "Overview of cloudlet, fog computing, edge computing, and dew computing," in *Proceedings of The 3rd International Workshop on Dew Computing*, 2018, pp. 20–23.

[33] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks*, vol. 130, pp. 94–120, 2018.

[34] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," in *2021 8th IEEE international conference on cyber security and cloud computing (CSCloud)/2021 7th IEEE international conference on edge computing and scalable cloud (EdgeCom)*. IEEE, 2021, pp. 139–142.

[35] G. Carvalho, B. Cabral, V. Pereira, and J. Bernardino, "Edge computing: current trends, research challenges and future directions," *Computing*, vol. 103, pp. 993–1023, 2021.

[36] M. Talebkhah, A. Sali, M. Marjani, M. Gordan, S. J. Hashim, and F. Z. Rokhani, "Edge computing: architecture, applications and future perspectives," in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*. IEEE, 2020, pp. 1–6.

[37] H. Elazhary, "Internet of things (iot), mobile cloud, cloudlet, mobile iot, iot cloud, fog, mobile edge, and edge emerging computing paradigms: Disambiguation and research directions," *Journal of network and computer applications*, vol. 128, pp. 105–140, 2019.

[38] M. Y. Akhlaqi and Z. B. M. Hanapi, "Task offloading paradigm in mobile edge computing-current issues, adopted approaches, and future directions," *Journal of Network and Computer Applications*, vol. 212, p. 103568, 2023.

[39] A. Acheampong, Y. Zhang, X. Xu, and D. A. Kumah, "A review of the current task offloading algorithms, strategies and approach in edge computing systems," *Computer Modeling in Engineering & Sciences, 134 (1)*, pp. 35–88, 2023.

[40] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE communications surveys & tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[41] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131 543–131 558, 2019.

[42] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7457–7469, 2020.

[43] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A survey of recent advances in edge-computing-powered artificial intelligence of things," *IEEE Internet of Things Journal*, vol. 8, no. 18, pp. 13 849–13 875, 2021.

[44] H. Hua, Y. Li, T. Wang, N. Dong, W. Li, and J. Cao, "Edge computing with artificial intelligence: A machine learning perspective," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[45] S. Wang, Y. Zhao, J. Xu, J. Yuan, and C.-H. Hsu, "Edge server placement in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 127, pp. 160–168, 2019.

[46] Z. Wang, W. Zhang, X. Jin, Y. Huang, and C. Lu, "An optimal edge server placement approach for cost reduction and load balancing in intelligent manufacturing," *The Journal of Supercomputing*, vol. 78, no. 3, pp. 4032–4056, 2022.

[47] Y. Gong, "Optimal edge server and service placement in mobile edge computing," in *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, vol. 9. IEEE, 2020, pp. 688–691.

[48] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, and W. Dou, "Complementing iot services through software defined networking and edge computing: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1761–1804, 2020.

[49] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, 2017.

[50] A. Wang, Z. Zha, Y. Guo, and S. Chen, "Software-defined networking enhanced edge computing: A network-centric survey," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1500–1519, 2019.

[51] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2017, pp. 94–100.

[52] R. Mogi, T. Nakayama, and T. Asaka, "Load balancing method for iot sensor system using multi-access edge computing," in *2018 Sixth International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE, 2018, pp. 75–78.

[53] A. Chandak and N. K. Ray, "A review of load balancing in fog computing," in *2019 International Conference on Information Technology (ICIT)*. IEEE, 2019, pp. 460–465.

[54] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[55] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2017.

[56] Y. Ai, M. Peng, and K. Zhang, "Edge computing technologies for internet of things: a primer," *Digital Communications and Networks*, vol. 4, no. 2, pp. 77–86, 2018.

[57] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[58] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE international conference on smart cloud (SmartCloud)*. IEEE, 2016, pp. 20–26.

[59] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[60] J. Cao, Q. Zhang, W. Shi, J. Cao, Q. Zhang, and W. Shi, "Challenges and opportunities in edge computing," *Edge Computing: A Primer*, pp. 59–70, 2018.

[61] R. Singh, R. Sukapuram, and S. Chakraborty, "A survey of mobility-aware multi-access edge computing: Challenges, use cases and future directions," *Ad Hoc Networks*, vol. 140, p. 103044, 2023.

[62] M. O'Grady, D. Langton, and G. O'Hare, "Edge computing: A tractable model for smart agriculture?" *Artificial Intelligence in Agriculture*, vol. 3, pp. 42–51, 2019.

[63] P. K. Malik, R. Sharma, R. Singh, A. Gehlot, S. C. Satapathy, W. S. Alnumay, D. Pelusi, U. Ghosh, and J. Nayak, "Industrial internet of things and its applications in industry 4.0: State of the art," *Computer Communications*, vol. 166, pp. 125–139, 2021.

[64] C. Sandu and I. Susnea, "Edge computing for autonomous vehicles-a scoping review," in *2021 20th RoEduNet Conference: Networking in Education and Research (RoEduNet)*. IEEE, 2021, pp. 1–5.

[65] L. U. Khan, I. Yaqoob, N. H. Tran, S. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge-computing-enabled smart cities: A comprehensive survey," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10 200–10 232, 2020.

[66] Z. Chen, W. Hu, J. Wang, S. Zhao, B. Amos, G. Wu, K. Ha, K. Elgazzar, P. Pillai, R. Klatzky *et al.*, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, 2017, pp. 1–14.

[67] J. Ren, Y. He, G. Huang, G. Yu, Y. Cai, and Z. Zhang, "An edge-computing based architecture for mobile augmented reality," *IEEE Network*, vol. 33, no. 4, pp. 162–169, 2019.

[68] Ç. Dilibal, "Development of edge-iomt computing architecture for smart healthcare monitoring platform," in *2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*. IEEE, 2020, pp. 1–4.

[69] M. Asif-Ur-Rahman, F. Afsana, M. Mahmud, M. S. Kaiser, M. R. Ahmed, O. Kaiwartya, and A. James-Taylor, "Toward a heterogeneous mist, fog, and cloud-based framework for the internet of healthcare things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4049–4062, 2018.

[70] M. Ahmad, M. B. Amin, S. Hussain, B. H. Kang, T. Cheong, and S. Lee, "Health fog: a novel framework for health and wellness applications," *The Journal of Supercomputing*, vol. 72, pp. 3677–3695, 2016.

[71] A. Al Omar, M. Z. A. Bhuiyan, A. Basu, S. Kiyomoto, and M. S. Rahman, "Privacy-friendly platform for healthcare data in cloud based on blockchain environment," *Future generation computer systems*, vol. 95, pp. 511–521, 2019.

[72] T. Muhammed, R. Mehmood, A. Albeshri, and I. Katib, "Ubehealth: A personalized ubiquitous cloud and edge-enabled networked healthcare system for smart cities," *IEEE Access*, vol. 6, pp. 32 258–32 285, 2018.

[73] R. M. Abdelmoneem, A. Benslimane, E. Shaaban, S. Abdelhamid, and S. Ghoneim, "A cloud-fog based architecture for iot applications dedicated to healthcare," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE, 2019, pp. 1–6.

[74] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, 2018.

[75] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 624–634, 2020.

[76] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Bedgehealth: A decentralized architecture for edge-based iomt networks using blockchain," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11 743–11 757, 2021.

[77] M. Rajasekaran, A. Yassine, M. S. Hossain, M. F. Alhamid, and M. Guizani, "Autonomous monitoring in healthcare environment: Reward-based energy charging mechanism for iomt wireless sensing nodes," *Future Generation Computer Systems*, vol. 98, pp. 565–576, 2019.

[78] I. Azimi, A. Anzanpour, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, "Self-aware early warning score system for iot-based personalized healthcare," in *eHealth 360°: International Summit on eHealth, Budapest, Hungary, June 14-16, 2016, Revised Selected Papers.* Springer, 2017, pp. 49–55.

[79] G. Muhammad, S. M. M. Rahman, A. Alelaiwi, and A. Alamri, "Smart health solution integrating iot and cloud: A case study of voice pathology monitoring," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 69–73, 2017.

[80] S. He, B. Cheng, H. Wang, Y. Huang, and J. Chen, "Proactive personalized services through fog-cloud computing in large-scale iot-based healthcare application," *China Communications*, vol. 14, no. 11, pp. 1–16, 2017.

[81] P. Verma, R. Tiwari, W.-C. Hong, S. Upadhyay, and Y.-H. Yeh, "Fetch: a deep learning-based fog computing and iot integrated environment for healthcare monitoring and diagnosis," *IEEE Access*, vol. 10, pp. 12 548–12 563, 2022.

[82] Q. Wu, X. Chen, Z. Zhou, and J. Zhang, "Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring," *IEEE Transactions on Mobile Computing*, vol. 21, no. 8, pp. 2818–2832, 2020.

[83] A. Singh and K. Chatterjee, "Edge computing based secure health monitoring framework for electronic healthcare system," *Cluster Computing*, vol. 26, no. 2, pp. 1205–1220, 2023.

[84] M. A. Rahman and M. S. Hossain, "An internet-of-medical-things-enabled edge computing framework for tackling covid-19," *IEEE Internet of Things Journal*, vol. 8, no. 21, pp. 15 847–15 854, 2021.

[85] O. S. Alwan and K. Prahald Rao, "Dedicated real-time monitoring system for health care using zigbee," *Healthcare technology letters*, vol. 4, no. 4, pp. 142–144, 2017.

[86] G. Aceto, V. Persico, and A. Pescapé, "The role of information and communication technologies in healthcare: taxonomies, perspectives, and challenges," *Journal of Network and Computer Applications*, vol. 107, pp. 125–154, 2018.

[87] M. Pham, Y. Mengistu, H. Do, and W. Sheng, "Delivering home healthcare through a cloud-based smart home environment (coshe)," *Future Generation Computer Systems*, vol. 81, pp. 129–140, 2018.

[88] M. Z. Uddin, "A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system," *Journal of Parallel and Distributed Computing*, vol. 123, pp. 46–53, 2019.

[89] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad, "Edge computing for smart health: Context-aware approaches, opportunities, and challenges," *IEEE Network*, vol. 33, no. 3, pp. 196–203, 2019.

[90] H. Yan, M. Bilal, X. Xu, and S. Vimal, "Edge server deployment for health monitoring with reinforcement learning in internet of medical things," *IEEE Transactions on Computational Social Systems*, 2022.

[91] A. M. Jasim and H. Al-Raweshidy, "Towards a cooperative hierarchical healthcare architecture using the hman offloading scenarios and srt calculation algorithm," *IET Networks*, vol. 12, no. 1, pp. 9–26, 2023.

[92] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2015.

[93] L. Zhao, W. Sun, Y. Shi, and J. Liu, "Optimal placement of cloudlets for access delay minimization in sdn-based internet of things networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1334–1344, 2018.

[94] Z. Xu, W. Liang, W. Xu, M. Jia, and S. Guo, "Efficient algorithms for capacitated cloudlet placements," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 2866–2880, 2015.

[95] Q. Fan and N. Ansari, "Cost aware cloudlet placement for big data processing at the edge," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[96] J. Meng, W. Shi, H. Tan, and X. Li, “Cloudlet placement and minimum-delay routing in cloudlet computing,” in *2017 3rd international conference on big data computing and communications (BIGCOM)*. IEEE, 2017, pp. 297–304.

[97] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, “Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 16, p. e3975, 2017.

[98] A. Santoyo-González and C. Cervelló-Pastor, “Network-aware placement optimization for edge computing infrastructure under 5g,” *IEEE access*, vol. 8, pp. 56 015–56 028, 2020.

[99] D. Li, C. Asikaburu, B. Dong, H. Zhou, and S. Azizi, “Towards optimal system deployment for edge computing: a preliminary study,” in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–6.

[100] Y. Li and S. Wang, “An energy-aware edge server placement algorithm in mobile edge computing,” in *2018 IEEE International conference on edge computing (EDGE)*. IEEE, 2018, pp. 66–73.

[101] T. Lähderanta, T. Leppänen, L. Ruha, L. Lovén, E. Harjula, M. Ylianttila, J. Riekki, and M. J. Sillanpää, “Edge computing server placement with capacitated location allocation,” *Journal of Parallel and Distributed Computing*, vol. 153, pp. 130–149, 2021.

[102] L. Lovén, T. Lähderanta, L. Ruha, T. Leppänen, E. Peltonen, J. Riekki, and M. J. Sillanpää, “Scaling up an edge server deployment,” in *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE, 2020, pp. 1–7.

[103] D. Bhatta and L. Mashayekhy, “A bifactor approximation algorithm for cloudlet placement in edge computing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1787–1798, 2021.

[104] Z. Wang, F. Gao, and X. Jin, “Optimal deployment of cloudlets based on cost and latency in internet of things networks,” *Wireless Networks*, vol. 26, no. 8, pp. 6077–6093, 2020.

[105] F. Zeng, Y. Ren, X. Deng, and W. Li, "Cost-effective edge server placement in wireless metropolitan area networks," *Sensors*, vol. 19, no. 1, p. 32, 2018.

[106] 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 *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC).* IEEE, 2017, pp. 683–690.

[107] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4568–4578, 2018.

[108] Z. Ning, X. Wang, J. J. Rodrigues, and F. Xia, "Joint computation offloading, power allocation, and channel assignment for 5g-enabled traffic management systems," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 5, pp. 3058–3067, 2019.

[109] A. Cabrera, A. Acosta, F. Almeida, and V. Blanco, "A dynamic multi–objective approach for dynamic load balancing in heterogeneous systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 10, pp. 2421–2434, 2020.

[110] G. Li, Y. Yao, J. Wu, X. Liu, X. Sheng, and Q. Lin, "A new load balancing strategy by task allocation in edge computing based on intermediary nodes," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–10, 2020.

[111] M. Z. Nayyer, I. Raza, S. A. Hussain, M. H. Jamal, Z. Gillani, S. Hur, and I. Ashraf, "Lbro: Load balancing for resource optimization in edge computing," *IEEE Access*, vol. 10, pp. 97 439–97 449, 2022.

[112] X. He, Z. Ren, C. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles," *China Communications*, vol. 13, no. Supplement2, pp. 140–149, 2016.

[113] 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. 99 658–99 669, 2019.

[114] P. P. Shahrbabaki, R. W. Coutinho, and Y. R. Shayan, "A novel sdn-enabled edge computing load balancing scheme for iot video analytics," in *GLOBECOM 2022-2022 IEEE Global Communications Conference*. IEEE, 2022, pp. 5025–5030.

[115] 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 of Things Journal*, 2023.

[116] R. Sikarwar, P. Yadav, and A. Dubey, "A survey on iot enabled cloud platforms," in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2020, pp. 120–124.

[117] H. N. Al-Anbagi and I. Vertat, "Cooperative reception of multiple satellite downlinks," *Sensors*, vol. 22, no. 8, p. 2856, 2022.

[118] A. Makkar, U. Ghosh, and P. K. Sharma, "Artificial intelligence and edge computing-enabled web spam detection for next generation iot applications," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 25 352–25 361, 2021.

[119] Lionel Sujay Vailshery. (2023) Iot connected devices worldwide. Accessed on 2023-11-08. [Online]. Available: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/

[120] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare internet of things: a survey of emerging technologies," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1121–1167, 2020.

[121] Facts and Factors, "Internet of things (iot) market size, share, trends - global forecast to 2028." [Online]. Available: https://www.fnfresearch.com/global-internet-of-things-iot-market-by-software-792

[122] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino, and A. Liotta, "An edge-based architecture to support efficient applications for healthcare industry 4.0," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 481–489, 2018.

[123] S. Javaid, S. Zeadally, H. Fahim, and B. He, "Medical sensors and their integration in wireless body area networks for pervasive healthcare delivery: A review," *IEEE Sensors Journal*, vol. 22, no. 5, pp. 3860–3877, 2022.

[124] Y. Zhao, W. Wang, Y. Li, C. C. Meixner, M. Tornatore, and J. Zhang, "Edge computing and networking: A survey on infrastructures and applications," *IEEE Access*, vol. 7, pp. 101 213–101 230, 2019.

[125] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[126] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[127] M. Stafford, A. Steventon, R. Thorlby, R. Fisher, C. Turton, and S. Deeny, *Briefing: Understanding the health care needs of people with multiple health conditions*. Health Foundation London, 2018.

[128] K. S. Yarnall, K. I. Pollak, T. Østbye, K. M. Krause, and J. L. Michener, "Primary care: is there enough time for prevention?" *American journal of public health*, vol. 93, no. 4, pp. 635–641, 2003.

[129] T. Østbye, K. S. Yarnall, K. M. Krause, K. I. Pollak, M. Gradison, and J. L. Michener, "Is there time for management of patients with chronic diseases in primary care?" *The Annals of Family Medicine*, vol. 3, no. 3, pp. 209–214, 2005.

[130] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[131] C. Lakshmi and S. A. Iyer, "Application of queueing theory in health care: A literature review," *Operations research for health care*, vol. 2, no. 1-2, pp. 25–39, 2013.

[132] L. Kleinrock, *Queueing systems*. Wiley, 1976.

[133] Z. Yang, B. Liang, and W. Ji, "An intelligent end–edge–cloud architecture for visual iot-assisted healthcare systems," *IEEE Internet of Things Journal*, vol. 8, no. 23, pp. 16 779–16 786, 2021.

[134] F. Wu, C. Qiu, T. Wu, and M. R. Yuce, "Edge-based hybrid system implementation for long-range safety and healthcare iot applications," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 9970–9980, 2021.

[135] A. Gutierrez-Torre, K. Bahadori, W. Iqbal, T. Vardanega, J. L. Berral, D. Carrera *et al.*, "Automatic distributed deep learning using resource-constrained edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 16, pp. 15 018–15 029, 2021.

[136] S. Oueida, Y. Kotb, M. Aloqaily, Y. Jararweh, and T. Baker, "An edge computing based smart healthcare framework for resource management," *Sensors*, vol. 18, no. 12, p. 4307, 2018.

[137] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1505–1521, 2022.

[138] S. Ebneyousef and A. Shirmarz, "A taxonomy of load balancing algorithms and approaches in fog computing: a survey," *Cluster Computing*, pp. 1–22, 2023.

[139] C. S. M. Babou, D. Fall, S. Kashihara, Y. Taenaka, M. H. Bhuyan, I. Niang, and Y. Kadobayashi, "Hierarchical load balancing and clustering technique for home edge computing," *IEEE Access*, vol. 8, pp. 127 593–127 607, 2020.

[140] M. Kyryk, N. Pleskanka, M. Pleskanka, and P. Nykonchuk, "Load balancing method in edge computing," in *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET).* IEEE, 2020, pp. 978–981.

[141] H. Pydi and G. N. Iyer, "Analytical review and study on load balancing in edge computing platform," in *2020 Fourth international conference on computing methodologies and communication (ICCMC).* IEEE, 2020, pp. 180–187.

[142] T. A. Al-Janabi and H. S. Al-Raweshidy, "Optimised clustering algorithm-based centralised architecture for load balancing in iot network," in *2017 International symposium on wireless communication systems (ISWCS).* IEEE, 2017, pp. 269–274.

[143] 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 Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017.

[144] C. Tang, C. Zhu, N. Zhang, M. Guizani, and J. J. Rodrigues, "Sdn-assisted mobile edge computing for collaborative computation offloading in industrial

internet of things," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 253–24 263, 2022.

[145] J. Li, J. Cai, F. Khan, A. U. Rehman, V. Balasubramaniam, J. Sun, and P. Venu, "A secured framework for sdn-based edge computing in iot-enabled healthcare system," *IEEE Access*, vol. 8, pp. 135 479–135 490, 2020.

[146] M. Priyadarsini, J. C. Mukherjee, P. Bera, S. Kumar, A. Jakaria, and M. A. Rahman, "An adaptive load balancing scheme for software-defined network controllers," *Computer Networks*, vol. 164, p. 106918, 2019.

[147] A. M. Jasim and H. Al-Raweshidy, "Optimal intelligent edge-servers placement in the healthcare field," *IET Networks*, vol. 13, no. 1, pp. 13–27, 2024.

[148] A. Jasim and H. Al-Raweshidy, "An adaptive sdn-based load balancing method for edge/fog-based real-time healthcare systems," *IEEE Systems Journal*, 2024.