Optimizing Resource Allocation and Offloading for Long-Term Load Balancing Solutions in Fog Computing

by Hamza H. Sulimani

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Dr Mukesh Prasad

University of Technology Sydney
Faculty of Engineering and Information Technology

January 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Hamza H. Sulimani, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy in the School of Computer Science, in the Faculty of Engineering, and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature:

Production Note:
Signature removed prior to publication.

Date: 06/11/2023

ACKNOWLEDGEMENTS

Allah, the Most Merciful and Kind, is worthy of all acclaim for allowing me to complete my dissertation and earn my doctorate. His grace has enabled me to accomplish all I have, but I am exclusively accountable for my failures.

Associate Professor Mukesh Prasad was an outstanding advisor throughout my PhD studies. I greatly appreciate your insightful comments and suggestions, which have substantially improved this study. Your commitment to your studies, generosity, and work ethic have left an indelible mark on me and will undoubtedly aid me in future endeavours.

My father has always supported and believed in me, and I owe her everything. He passed away before I began my doctoral studies. Please, Allah, let my father's essence enter paradise.

My primary goal is to express sincere appreciation to my mother. Regardless of my difficulties, she was always encouraging and supportive. My mother and I have set a lifetime goal to complete a doctoral program. I do not know if I would ever succeed if not for my mother's prayers, encouragement, vast understanding, and unwavering support. Consequently, I have the highest regard and affection for my mother.

Similarly, I would like to convey my appreciation to my relatives. I appreciate the Saudi Arabian Cultural Mission (SACM) in Australia, the Saudi Arabian Embassy in Australia, and Umm Al-Qura University.

LIST OF PUBLICATIONS

JOURNAL ARTICLES PUBLISHED

1. Sulimani, H., et al., *Reinforcement optimisation for decentralised service placement policy in IoT-centric fog environment.* Transactions on Emerging Telecommunications Technologies, 2022: p. e4650. **(Chapter 2)**

CONFERENCE PAPER PUBLISHED

1. Sulimani, H., et al., *Sustainability of Load Balancing Techniques in Fog Computing Environment.* Procedia Computer Science, 2021. **191**: p. 93-101. **(Chapter 3)**

TABLE OF CONTENTS

Contents

CERTIFICATE OF ORIGINAL AUTHORSHIP II
ACKNOWLEDGEMENTS III
LIST OF PUBLICATIONS IV
TABLE OF CONTENTS V
LIST OF FIGURES VIII
LIST OF TABLES X
ABBREVIATIONS XI
ABSTRACT XII
1. CHAPTER, INTRODUCTION 2
1.1. BACKGROUND 3
1.2. RESEARCH ELEMENTS 6
1.2.1. RESEARCH PROBLEMS 6
1.2.2. RESEARCH QUESTIONS 9
1.2.3. RESEARCH OBJECTIVES 9
1.2.4. RESEARCH CONTRIBUTIONS 12
1.3. RESEARCH METHODOLOGY 13
1.3.1. GENERAL METHODOLOGY 13
1.3.2. RESEARCH PLAN 15
1.4. RESEARCH FRAMEWORKS 20
1.4.1. THEORETICAL FRAMEWORK 20
1.4.2. CONCEPTUAL FRAMEWORK 23
1.5. THESIS STRUCTURE 24
1.6. PUBLICATION RELATED TO THIS RESEARCH 25
2. CHAPTER, LITERATURE REVIEW 26
2.1. INTRODUCTION 27
2.2. RELATED REVIEW STUDIES 29
2.2.1. THE MOTIVATIONS FOR THE SYSTEMATIC REVIEW 30
2.2.2. RESEARCH CONTRIBUTION 31
2.3. RESEARCH METHOD 31
2.3.1. PLANNING REVIEW 32
2.3.2. CONDUCTING AND DOCUMENTING THE REVIEW 34
2.4. BACKGROUND OF RESEARCH 35

2.4.1. Fog Computing (FC) 36
2.4.2. Load Balancing (LB) 44
2.5. Challenges in Fog Computing 49
2.5.1. Service-Oriented 50
2.5.2. Structural Issues 50
2.5.3. Security Aspects 50
2.6. Discussion 50
2.7. Conclusion 53
3. Chapter, Reinforcement Optimisation for Decentralised Service Placement Policy in IoT-Centric Fog Environment 54
3.1. Introduction 55
3.1.1. Research Problem 56
3.2. Related Works 57
3.3. Proposed Architecture and Algorithm 60
3.3.1. Architecture 60
3.3.2. System Model 65
3.3.3. Optimisation Model 66
3.4. Evaluation and Experimental Results. 70
3.4.1. Evaluation 70
3.4.2. Experimental Results 72
3.5. Discussion 74
3.6. Conclusion 78
4. Chapter, HybOff: A Hybrid Approach to Improve Load Balancing in Fog Networks 79
4.1. Introduction 80
4.2. Literature Review 83
4.2.1. Fog Computing 83
4.2.2. Related Works 84
4.3. Hybrid Approach To Enhance Load Balancing 88
4.3.1. Problem Formulation and Terminology 90
4.3.2. HybOff Design 93
4.3.3. Hybrid Framework 94
4.3.4. The Proposed Algorithm 100
4.4. Experiments And Results 101
4.4.1. Preliminary Explanations 101
4.4.2. Environment Description 102

4.5. Discussion 107
4.6. Stage 1c, Optimum LB Solution 109
4.6.1. Discussion 112
4.6.2. Limitations 113
4.7. Conclusion. 115
5. Chapter, A Comprehensive Solution: A Sustainable Load Balancing Monitoring System for the Hidden Costs in Fog Computing 116
5.1. INTRODUCTION 117
5.2. BACKGROUND OF THE RESEARCH AREA 119
5.2.1. Sustainability 119
5.2.2. Reinforcement Learning (RL) 120
5.2.3. Load Balancing in a Fog Environment 121
5.2.4. Related Works 121
5.3. Research Problem 122
5.3.1. The Prevalent Solution 123
5.3.2. Exploring the new direction of research 125
5.4. The Proposed Sustainable Load-Balancing Monitoring System (SLBMS) 126
5.4.1. The Architecture 127
5.4.2. The Structure Of The Model 129
5.5. Implementation and evaluation 134
5.5.1. Used Access Points Dataset 134
5.5.2. A Mathematical RL Illustration Model 135
5.5.3. Evaluation metrics 137
5.5.4. SLBMS implementation 138
5.6. Discussion 139
5.7. Conclusion 141
6. Chapter, Conclusion and Future Research 142
6.1. Introduction 143
6.1.1. Research Overview 143
6.1.2. Research Phases 146
6.2. Load Balancing Solutions 147
6.3. Model Limitations in the real world 148
6.4. Future Works 150
Bibliography 151
Appendix 162

LIST OF FIGURES

FIGURE 1.1: NUMBER OF CONNECTED IOT DEVICES GLOBALLY. 5
FIGURE 1.2: VENN DIAGRAM FOR RESEARCH GAPS. 8
FIGURE 1.3: RESEARCH OBJECTIVES. 12
FIGURE 1.4: FUNDAMENTAL STAGES FOR RESEARCH METHODOLOGY. 14
FIGURE 1.5: THE ACTUAL IMPLEMENTATION OF THIS RESEARCH. 15
FIGURE 1.6: FIFO BEHAVIOR IN THE PROPOSED MODEL. 17
FIGURE 1.7: DEALING WITH THE LIGHT TASK IN THE QUEUE. 18
FIGURE 1.8: CLUSTERING IN THE PROPOSED ALGORITHM. 18
FIGURE 1.9: THE PROPOSED SOLUTION FOR THE RESEARCH PROBLEM. 20
FIGURE 1.10: THE RAPPORT AMONG VARIABLES OF RESEARCH. 21
FIGURE 1.11: THE CONCEPTUAL FRAMEWORK OF THE STUDY. 23
FIGURE 1.12: THESIS ORGANIZATION. 24

FIGURE 2.1: THE STAGES OF THE RESEARCH METHODOLOGY. 32
FIGURE 2.2: FOG ARCHITECTURE. 39
FIGURE 2.3: OPENFOG ARCHITECTURE. 41
FIGURE 2.4: TYPE OF CONTROLLER IN FOG LAYER. 48
FIGURE 2.5: POSSIBLE OFFLOADING SCENARIOS IN A FOG ENVIRONMENT. 49

FIGURE 3.1: INTEROPERATED SERVICES AND APPLICATIONS. 61
FIGURE 3.2: EXAMPLE OF SERVICE MIGRATION WITHIN DIFFERENT PATHS. 63
FIGURE 3.3: EXAMPLE OF INTEROPERATED SERVICE MIGRATION WITHIN THE SHORTEST PATH. 64
FIGURE 3.4: DECENTRALISED SERVICE PLACEMENT, THE BROKER 65
FIGURE 3.5: PROPOSED NETWORK. 71
FIGURE 3.6: APPLICATION EDGES FOR ONLINE STORE-BASED CASE STUDY. 72
FIGURE 3.7: HOPS COUNT WITH DIFFERENT SITUATIONS. EXPERIMENT WITH TWO APPLICATIONS, TWO USERS, AND TWO LEVELS OF FOG DEVICES: (A) CHANGING THE NUMBER OF DEVICES ASSOCIATED WITH THE EDGE DEVICE AND (B) CHANGING THE NUMBER OF PROPOSED FOG LEVELS. 75
FIGURE 3.8: SERVICE LATENCY IN DIFFERENT SITUATIONS. EXPERIMENT WITH TWO APPLICATIONS, TWO USERS, AND TWO LEVELS OF FOG DEVICES: (A). ... 76
FIGURE 3.9: CPU USAGE OF THE DEVICES REGARDING THEIR TOPOLOGY DISTRIBUTION. EXPERIMENT WITH TWO APPLICATIONS AND TWO USERS... 77

FIGURE 4.1: PREVALENT OFFLOADING, COSTS, AND SOLUTIONS. 90
OBJ
OBJ
FIGURE 4.4: STATIC OFFLOADING TABLE IN THE MASTER SERVER. 96
FIGURE 4.5: HYBOFF PROCESS FLOWCHART. 100
FIGURE 4.6: BALANCE OF RESOURCE UTILIZATION-MATHEMATICAL, (A) BEFORE OFFLOADING, (B) SORTED SERVERS, (C) PAIRED SERVERS, AND (D) AFTER OFFLOADING. 101

FIGURE 4.7: RESOURCE UTILIZATION OVER TIME AND CHANGING NUMBER OF SERVERS. (A) RESOURCE UTILIZATION OF THE SYSTEM WITH CHANGING OF TIME, AND (B) RESOURCE UTILIZATION PERCENTAGE.....104
FIGURE 4.8: THE PERCENTAGE OF HEALTHY SERVERS FOR THE THREE ALGORITHMS WITH THE SAME MEAN VALUE BUT DIFFERENT STANDARD DEVIATIONS, WHERE (A) IS THE PERCENTAGE OF BALANCED SERVERS, AND (B) IS THE STANDARD DEVIATION FOR ALGORITHMS.....106
FIGURE 4.9: TSAS PERFORMANCE EVALUATION WITH (A) DIFFERENT DATA SIZES AND (B) DIFFERENT SYSTEM SCALES.....107
FIGURE 4.10: THE DEFINITION OF CELL IN MODELS.....110
FIGURE 4.11: CPU USAGE OF THE DEVICES ABOUT THEIR TOPOLOGY DISTRIBUTION. EXPERIMENT WITH TWO APPLICATIONS.....111
FIGURE 4.12: HOPS COUNT WITH DIFFERENT SITUATIONS. EXPERIMENT WITH TWO APPLICATIONS, TWO USERS, AND THREE FOG LEVELS: (A) CHANGING THE NUMBER OF GENERATED TASKS. 111
FIGURE 4.13: THE LEVEL OF LB FOR THE PROPOSED MODELS UNDER DIFFERENT SIZE OF GENERATED TASKS AND CONSTANT COMPUTING POWER. 112

FIGURE 5.1: PREVALENT OFFLOADING PROCESS FLOWCHART.....124
FIGURE 5.2: POSSIBLE SOLUTIONS FOR THE PROBLEM OF RESEARCH.....125
FIGURE 5.3: THE PROPOSED MODEL ARCHITECTURE. 127
FIGURE 5.4: THE EFFECT OF RPQ-STAGE ONE.....128
FIGURE 5.5: TYPE OF EXCHANGED DATA IN APS.....130
FIGURE 5.6: SLBMS MODEL.....133
FIGURE 5.7: AVERAGE WT IN COMPARISON.....138
FIGURE 5.8: THE NUMBER OF EXCHANGED TASKS IN THE SPECIAL CASE OF HYBOFF AND AFTER ENGAGING SLBMS IN THE SOLUTION.....139

LIST OF TABLES

TABLE 1.1: THE PLANNING OF THESIS PUBLICATIONS. 25

TABLE 2.1: THE USED DATABASES. 30
TABLE 2.2: A COMPARISON AMONG DIFFERENT SYSTEMATIC STUDIES OF LB APPROACHES IN THE FC. 30
TABLE 2.3: THE EXCLUSION AND INCLUSION CRITERIA. 34
TABLE 2.4: SEARCHING PROCESS. 35
TABLE 2.5: THE DIFFERENCES BETWEEN FOG AND CLOUD COMPUTING. 44

TABLE 3.1: SUMMARY OF APPROACHES TO SERVICE PLACEMENT 60
TABLE 3.2: SUMMARY OF THE FUNCTIONS AND VARIABLES USED IN THE PROPOSED MODEL. 67
TABLE 3.3: THE SUMMARY OF CONFIGURATIONS OF EXPERIMENTS. 70
TABLE 3.4: SYSTEM SPECIFICATION 72

TABLE 4.1: ESSENTIAL NOTATIONS. 91
TABLE 4.2: FEATURES OF HYBOFF MODULES. 94
TABLE 4.3: INITIAL PARAMETERS OF EXPERIMENT. 102
TABLE 4.4: SPECIFICATIONS OF FOG SERVERS. 103
TABLE 4.5: TASK SPECIFICATIONS. 103

TABLE 5.1: DEVICES ATTRIBUTES TABLE (DAT). 132
TABLE 5.2: STATES OF THE CELLS OF THE SYSTEM. 133
TABLE 5.3: DA-TABLE OF THE SYSTEM. 135
TABLE 5.4: THE INITIAL VALUES OF THE PARAMETERS. 135
TABLE 5.5: Q-TABLE- INITIALISED. 135
TABLE 5.6: Q-TABLE-UPDATED. 136
TABLE 5.7: EVALUATION METRICS. 137
TABLE 5.8: WT AND TAT FOR HYBOFF IN DIFFERENT STATES 138

ABBREVIATIONS

IoT - Internet of Things
LB – Load Balancing
HybOff- Hybrid Offloading
SlbmS- Sustainable Load Balancing Monitoring System
RODSPP - Reinforcement Optimisation for a Decentralised Service Placement Policy.
RAN - Radio Access Network
ISG - Industry Specification Group
ETSI - European Telecommunications Standards Institute.
AWS - Amazon Web Services.
AR - Augmented Reality
IoV - Internet of Vehicles
VFC- Vehicular Fog Computing.
MFL- Monitoring Fog Layer
SOT- Static Offloading Table
TADF- Top Affected Device Factor
CDN - Content Delivery Network
LBM- Load Balancing Module
RM- Recommendation System
AP - Access Point
QoS - Quality of Service
QoE - Quality of Experience
DRL - Deep Reinforcement Learning
RSS - Radio Signal Strength.
SEC - Symposium on Edge Computing.
SPRM - Service placement request manager

ABSTRACT

OPTIMISING RESOURCE ALLOCATION AND OFFLOADING FOR LONG-TERM LOAD BALANCING SOLUTIONS IN FOG COMPUTING ENVIRONMENTS

by

Hamza H. Sulimani

Nowadays, most emerging critical IoT applications have unique requirements and restrictions to operate efficiently; otherwise, they could be useless. Latency is one of these requirements. Fog computing is the complement system for cloud computing, proving it is the ideal computing environment for critical IoT applications.

Distributed computing systems, such as fog computing, have an inherent problem when the computing units have different computing loads, called load difference problems. Offloading and service placement are some techniques used to fix these problems. Although prevalent offloading is the appropriate technique for this research, its procedures generate hidden costs in a system, such as decision time, distant offloading, and network congestion. Many researchers attempt to reduce these costs to get the results of static offloading (in stable environments).

However, this research seeks to overcome the hidden costs in the prevalent offloading techniques to balance the load in a fog environment by utilising the sustainability concept. This research believes that increasing physical resources is the only way to improve efficiency as a long-term solution. The study consists of two consecutive phases. The first phase attempts to find the optimum solution between task offloading and service placement. The solution must revive the low-cost offloading solution. A Sustainable load-balancing monitoring system (SlbmS) is the comprehensive solution for the optimum solution to release its limitations. SlbmS uses the sustainability concept to solve the problem of the limitation of resources in edge computing using reinforcement learning.

The experiment results of the two phases show that hybrid offloading outperforms the service placement policy in the first stage and prevalent offloading in the second stage when utilising the behaviour of static offloading to reduce the offloading costs in unpredictable environments. The study aims to explore a new area of research that attempts to amend the network topology to improve resource provisioning to provide a free resource at the edge of the network. This research paved the way for a new dimension of analysis. It is the first research to recommend the physical expansion in the fog layer using the sustainability concept.

THE CHAPTERS

1. CHAPTER, INTRODUCTION

Abstract: The success of any project depends mainly on its planning process. In a PhD journey, an average PhD candidate spends around one year exploring and planning potential research gaps for his thesis. A well-planned PhD research study should adhere to a specific research plan and framework. A research plan must be discussed and approved to hold the research; each research's objectives, concerns, and aims serve as its foundation. Fog computing is the leading research area of this thesis's investigation. This chapter, **however,** attempts to sketch out the foundations of the study of new dimensions for balancing the load among the computing units in the fog layer. Offloading and resource allocation are the core elements to balance the load in a fog environment. The outcomes of this chapter are the research plan framework and thesis structure followed in the following years after being identified and approved. The procedure for conducting the research is detailed in this chapter.

1.1. BACKGROUND

In the era of the internet, with the support of 5G access networks, the central computing concept has appeared primarily in all fields. The central computing system is the technology that allows enterprises to collect, process, analyse, and archive the distributed client's data worldwide. The main concept has countless benefits for enterprises, such as easy management, protection, and timely business updates (Traub, Maier et al. 2017). The idea has become so important with the internet that we can no longer return to primitive, decentralised systems.

On the ground, cloud computing is implementing the central computing concept. It offers novel possibilities for both service providers and their clients through an architecture for delivering Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) on-demand over the internet that promises to bring about significant economic and technical benefits (Celesti, Fazio et al. 2012). These large-scale services can be provided using virtualised cloud resources on a shared basis. Cloud resources are generally made available as a compilation of several proprietary processes running in a virtual environment known as a Virtual Machine (VM).

In a cloud environment, virtualised computational resources are used to provision resources on demand. Virtualization also enables an auto-scaling technique that dynamically allocates computational resources to services that precisely correspond to their current needs, thereby eliminating inactive and costly resources (Kriushanth, Arockiam et al. 2013). Advances in virtualisation techniques and the construction of numerous large commodity data centres around the globe have resulted in cloud computing, a new approach to computing, becoming a significant research and development topic (Ghanbari, Simmons et al. 2012). The recent increase in the prevalence and utilisation of cloud computing services by enterprise and individual consumers necessitates the efficient and proactive administration of data centre resources that host services with diverse characteristics.

Cloud Computing is indispensable for any business to connect and communicate directly with its intelligent products to manage its business

proficiently. The cloud computing model has been widely appreciated as appropriate machinery to efficiently run/manage widely distributed IoT (Internet of Things) products. IoT is a telecommunication system that enables endless processing devices to exchange data over a public network to keep a business running with minimal human intervention (Sadeeq, Abdulkareem et al. 2021); it is considered a system tool that allows cloud computing to interact with the environment. Currently, most of the surrounding intelligent products, which surmount daily hurdles, are enabled by the IoT, meaning they may only fully operate with an internet connection (Aazam, Zeadally et al. 2018). This setup allows IoT technology to spread widely and its data to grow gradually, which gives the cloud system more popularity while, on the other hand, affecting the efficiency of public networks.

When the public network, the primary channel for the cloud to work, slows down, time-sensitive applications are the ones that suffer the most. Time-sensitive applications are characterised by timing constraints that must be satisfied for correct operation (Goel, Abeni et al. 2002). They must be served within a reasonable time limit to be more effective and work well. Noteworthy, all IoT applications use the same public network; public networks are designed to serve all applications without bias. Hence, providers of time-sensitive applications must consider this when introducing critical services, such as e-health, smart grids, and unmanned vehicles (Ruan, Guo et al. 2021, Gupta and Gupta 2022).

However, cloud computing cannot offer the required level of service for this type of application due to the uncontrolled efficiency of public networks (Wójcicki, Biegańska et al. 2022). Fog Computing (FC) has been proposed as a critical technology to provide decentralised computation services in decentralised settings, which would help address the challenges in cloud computing and deal with the problems listed above. FC is designed to operate IoT applications efficiently, especially real-time applications (Gomes, Costa et al. 2021).

The continued growth in the number of IoT objects and the data generated means FC cannot deliver the expected performance level (Sabireen and Neelanarayanan 2021). Due to the widespread use of mobile devices and the rapid advancement of wireless networking technologies over the past ten years, mobile

users' demand for computation has increased unprecedentedly. According to Statista. (2023), the total number of connected devices will exponentially increase to around thirty billion devices in 2030, Figure 1.1, which means that the amount of data generated by these devices will be uncontrollable. On this basis, the strict QoS requirements for computation-intensive and latency-critical applications severely threaten distributed processing models, including FC (Satka, Ashjaei et al. 2023).

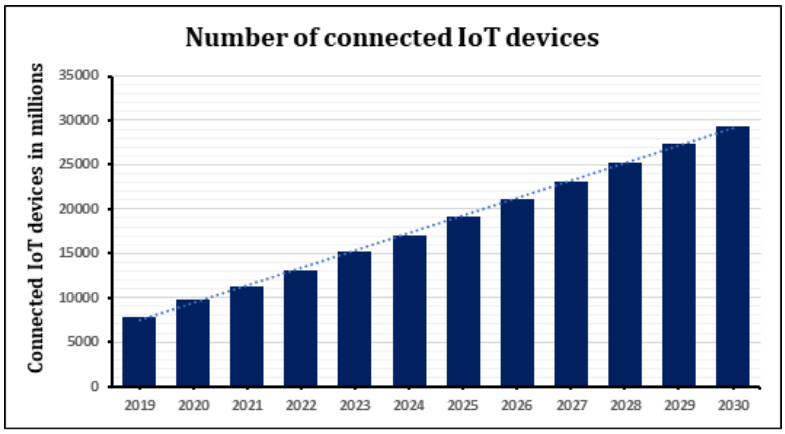


Figure 1.1: Number of connected IoT devices globally.

As a result, the unexpected growth in the number of IoT devices has different rates in each area. Due to the differences in the growth rate of user density in coverage, some fog devices reach an overloaded state while others remain idle. Moreover, the unpredictability of cloud service/customer demands causes the workload of VMs to vacillate dynamically, resulting in an imbalance in both the burden and utilisation of virtual and physical cloud resources. These standing increases resource wastage and causes misalignment in the fog layer. Much research has proven that Load Balancing (LB) and resource allocation can solve this situation (Sofla, Kashani et al. 2022).

As was broadly discussed in the prior research, producing a balanced fog system through a suitable resource allocation policy is necessary to reinforce the fog environment to work efficiently with sensitive applications (Sulimani, Alghamdi et al. 2021). LB is a process that allows the computing units to exchange the workloads among layers to fill clients' needs regarding resources with the least possible delay (Kashani, Ahmadzadeh et al. 2020). This research considers offloading as the appropriate technique to allocate resources in FC. However,

offloading costs the computing system in terms of time and network. Therefore, providing a powerful solution for offloading is vital, allowing cloud technology via FC to continue running well to serve real-time applications.

However, LB optimisation problems via dynamic offloading have been extensively studied; most proposed optimisation models focused on improving dynamic offloading, which has an inherent cost and follows a short-term theme. To the best of our knowledge, they need to use the concept of long-term in their optimisation model.

Considering these facts, this study prepares a combination of the three main topics of LB as a core system. It proposes a reinforcement optimisation for a decentralised service placement policy in fog environments (RODSPP) and novel techniques for hybrid offloading (HybOff). Furthermore, a comprehensive solution named a sustainable load balancing monitoring system (SlbmS) for LB has been developed in this study to overcome the drawbacks of current solutions.

The remaining part of this chapter is structured as shown next. Section 2 presents the research description, including the research problems, objectives, and expected contributions. In Section 3, the research methodology and plan are described. Section 4 outlines the thesis frames, while Section 5 presents the structure. The publications of research are enumerated in Section 6.

1.2. Research Elements

This section presents the major research elements that outline the research borders, such as motivations, challenges, and problem statements. Moreover, the gaps, questions, objectives, and research methodology are listed here.

1.2.1. Research Problems

Many problems are motivated to start working in this area of research. The following issues have been noticed during the review stage. Some inherent costs were found in the prevalent offloading process. This subsection illustrates the primary issues that motivate this study, as listed next:

1) **Offloading Costs.** Offloading is a crucial technique in the LB process. According to our observation, we have noticed that all prevalent offloading

approaches use the present system state information, in which the heavy devices read the environment (gathering attributes) and give an offloading action to redirect the excess tasks to the target device (Jaya 2023). This action is repeated several times when there is a necessity for extra resources. There are multi-costs due to offloading activities, such as:

a. **Decision time**. Most prevalent offloading models have many procedures, such as gathering, analysing, and evaluating system attributes, to take the action required. These procedures might consume an unnoticeable time to find a suitable device for offloading. Even though this time is minimal, it repeats many times in every loaded device, which costs a system hefty of time. We call this time *a decision time.* This time accumulates, which increases the total execution time for the task.
b. **Decision messages.** Decision processes generate a high volume of exchanged messages with the peer devices; they seek to explore unused resources to cover the shortage in the affected areas without **any intention** to increase the number of served devices (resources) (Sofla, Kashani et al. 2022). We call these messages *decision messages*. Growing the number of exchanged ***decision*** **messages** consumes more time and causes network congestion.
c. **Network Oversizing.** However, it may only be the appropriate decision to expand the network if we are confident that all fog devices are fully equipped and utilised, especially with the apparent differences in the efficiency of LB algorithms. In this case, the expansion in network size generates a unique network state, which we propose to name ***network oversizing****.* More wastage in power and resources occurs if the offered resources exceed the demand; this state typically appears in small businesses where they **do not intend** to study or plan for their networks before expanding.
d. **Need for more planning.** A network oversizing state occurs when the network keeps expanding by enterprise without *sufficient planning*—the network oversizing conflicts with a sustainability concept that seeks to avoid significant or unreasonable physical

changes. Thus, adequate planning for the expanding action is required to keep the performance of FC at an acceptable level without incurring high costs.

e. **Distant offloading.** The other effect of network oversizing is distant offloading. It usually appears with the large-scale system. With this system, the overloaded device might offload its tasks to the device located away. While many offloading algorithms are not concerned with this point, it surges the consumed time with the exchanged messages.

2) **Time-sensitive applications.** Many emerging applications are real-time, which is more valuable and applicable to the users. Time-sensitive applications require a harsh criterion to run effectively. Thus, our research aims to enhance the quality of this type of application.

3) **Resource constraints.** Conceptually, once LB systems start offloading, the local fog resources are exhausted, and the workload increases substantially (Aazam, Zeadally et al. 2018, Sulimani, Sajjad et al. 2022). However, resource provisioning is required to find extra resources. Most existing resource provisioning research in FC uses granularity services such as virtual resource, container-based, application, and task provisioning services to enhance the capabilities of available local resources (Shakarami, Shakarami et al. 2022). However, these services are limited by physical resources and can only go up to that. Figure 1.2 illustrates the research gaps.

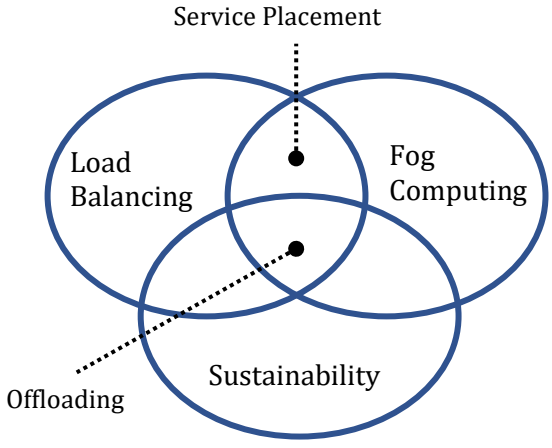


Figure 1.2: Venn diagram for research gaps.

1.2.2. Research Questions

Based on the problems mentioned above and the fact that both service placement and offloading techniques rely on offloading strategy, the research questions of this study are determined as follows:

- **Research Question 1:** What is the required technique to minimise the consumed time by offloading the decision process?
- **Research Question 2:** What is the required strategy to minimise the number of offloading messages?
- **Research Question 3:** What is the suitable strategy to avoid the distant offloading phenomena?
- **Research Question 4:** What is a new solution for optimal placement policy over fog physical resources to overcome the inherent issues of prevalent resource allocation mentioned in RQ1, RQ2, and RQ3?
- **Research Question 5:** What is a new solution for optimal offloading over fog physical resources to overcome the inherent issues of prevalent resource allocation mentioned in RQ1, RQ2, and RQ3?
- **Research Question 6:** How could the performance of the proposed systems be evaluated in a fog environment in RQ4 and RQ5?
- **Research Question 7:** What are the limitations of the optimum LB solution in this research?
- **Research Question 8:** How should reinforcement learning be designed to apply sufficient planning before expansion?
- **Research Question 9:** How do we design a comprehensive model to monitor the entire system and suggest a recommendation to tackle the overload situation by proposing hardware modifications?

1.2.3. Research Objectives

Generally, dynamic behaviour is typically required. Several offloading strategies have been applied by examining the prevalent solutions, which may reduce the load on some nodes but generate drawbacks. Most of these studies, however, ignore the significant impact on network bandwidth and the time needed for offloading decisions, which must improve the level of LB in the system. Based on the research problems and questions, this study has nine primary objectives to solve the dilemma of the research problem, as shown next:

Research Objective One. The first research objective related to the first research question is to propose a new offloading philosophy that avoids the consumed time in dynamic offloading to keep the time at the bottom. The new philosophy should

partially follow the static offloading philosophy. Static offloading is the primarily used system; it has many drawbacks, as mentioned in (Sulimani, Alghamdi et al. 2021). Both dynamic and static offloading have inherent issues. Even though the recent research focused on dynamic offloading while ignoring the static, adopting a new hybrid offloading process is necessary. The new process will cease the consequences of both approaches.

Research Objective Two. This objective, which corresponds to the second research question, is to propose a solution to minimise the number of decision messages in a system that aims to reduce the traffic in the network. The proposed solution must be acceptable with the quality of the present system state information, which is essential to evaluate the surrounding devices. This feature cannot be found in static offloading, where static offloading is not required to collect the present system state information.

Research Objective Three. The third research objective related to the third research question is to propose a technique to solve the distant offloading to alleviate the communication cost with the remote device, which might be located away. The proposed solution must keep the devices to communicate as a group.

Research Objective Four. The fourth research objective corresponding to question four is to design a new service placement policy to enhance the LB level in FC. After finding a suitable research gap, the new algorithm should reinforce one state-of-the-art algorithm. This stage aims mainly to explore another approach to balance the load in FC. At this point, we are going to introduce placement algorithms with the goals of decreasing latency, lightening the load on bandwidth, and improving the quality of service. This stage aims to understand the drawbacks of current service placement algorithms and create a practical model that enhances the efficiency of existing computing nodes as much as possible.

Research Objective Fifth. The primary goal of this proposal is to construct a hybrid offloading algorithm to maintain equilibrium for FC. It proposes offloading algorithms with the goals of decreasing latency, lightening the stress on bandwidth, and boosting the quality of service. This stage aims to understand the drawbacks of

current offloading algorithms and create a practical model that improves the efficiency of resource allocation policy as much as possible.

Research Objective Six. The sixth objective of the investigation is to evaluate the developed algorithms. Evaluation is a crucial component of every methodology because it provides reasonable assurance of the model's results. This research relies on three evaluation models and empirical environments to evaluate the performance of three new systems devised for this study, including:

- RODSPP
- HybOff
- SlbmS

The RODSPP and HybOff prototype systems are implemented in simulation environments and evaluated using iFogSim. SlbmS is evaluated by implementing its subsystems in MATLAB. This stage aims to determine the optimal solution to balance the load of fog nodes among two techniques.

Research Objective Seven. To identify the optimum solution, we need to find the constraints or limitations of this solution. The identified solution must be tested with different scenarios with different sizes of cells and generated tasks.

Research Objective Eight. The eighth objective of this research is to utilise reinforcement learning to satisfy the requirement of sufficient planning. The RL must work through two steps. In step one, the proposed model must study how to improve the coverage quality; in step two, the system must learn the efficiency of adding new Access points to the system. This technique gives a chance before engaging in an unnecessary expansion process.

Research Objective Nine. This aim aims to find a sustainable solution to the research dilemma. As we have noticed during the review stage that all prevalent solutions for LB work around the same notion, we aim to find a novel solution for this dilemma in this research. The primary motivation of this research is to determine how to build a framework to balance the load in fog environments, which is a long-term concept. Accordingly, this research intends to go beyond prevalent resource provisioning by exploring a new research dimension. However, Figure 1.3 depicts and illustrates the rapport among the research objectives.

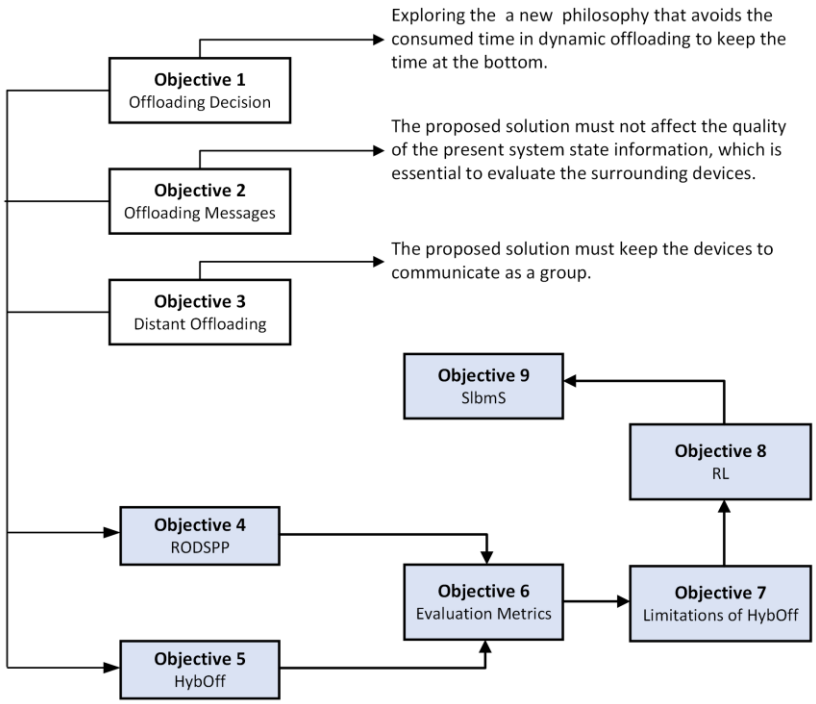


Figure 1.3: Research Objectives.

1.2.4. Research Contributions

According to the objectives of the study, the following innovations and contributions to research are summarised:

1) **Develop a service placement policy:** This research proposes a reinforcement service placement policy as a prototype solution to explore the other LB technique. It covers two major fog resource management topics: service migration and task offloading. It is the first technique in this research that attempts to balance the load. In FC, service placement or migration to Virtual Machines (VMs) affects the level of LB. On the other hand, task offloading should be optimised resource allocation, which improves the utilisation level. Therefore, the research considers service placement and task allocation to maximise the QoS for time-sensitive applications.
2) **Develop a novel hybrid offloading model:** It is the first work that attempts to apply the real meaning of hybrid offloading, which gains both the behaviour of static and dynamic offloading. The existing research in task offloading migrates the tasks from loaded devices to other non-heavy devices using different techniques. All prevalent task-offloading methods use the same pattern with hidden costs (Sulimani, Alghamdi et al. 2021). However, task offloading is affected by the unpredictable number of arrival tasks and

the immutable capacity of the computing unit. This model can overcome the inherent issues in static and dynamic offloading with outstanding results in terms of latency.

3) **Hidden costs:** It discussed hidden costs such as distant offloading, decision messages, and network congestion.
4) **Cease offloading processes:** While all prevalent LB models seek to enhance the efficiency of offloading techniques, this research's ultimate goal seeks to minimise or cease offloading procedures.
5) **Long-Term Solution:** This research aims to answer the main research question, "How to build a comprehensive model applying sustainable concepts for the research dilemma?" The goal will be secured by answering this part of the research. It demonstrates how to apply the sustainability concept to the proposed solution.
6) **Multi-feature system:** The proposed model contains many features, such as following a central-distributed control system, clustering fog devices, and prioritising critical applications.
7) **Experiments:** It presents the results of the three comprehensive experiments that examine the proposed models from different aspects.

1.3. Research Methodology

Research methodology is a collection of problem-solving techniques that follow a set of fundamentals and a shared philosophy for addressing specific issues (Gallupe 2007). Numerous research methodologies have been proposed and utilised in information systems, including design research, case study, field experiment, field study, laboratory experiment, action, and survey research. This study's methodology is based on the practice of design research (Niu, Lu et al. 2009), which has been proposed and implemented in information systems.

1.3.1. General Methodology

As shown in Figure 1.4, the design research methodology consists of five fundamental stages (Niu, Lu et al. 2009):

1. **Problem definition:** This is the initial step in identifying the limitations of existing applications and significant research problems. Existing applications must catch up to the anticipated status, reflected by the *research problems*.

Various sources, including industry experience, observations of practical applications, and literature review, can be used to identify research problems. A precise definition of the research problem focuses on the investigation throughout the development process. This phase produces a research proposal for new research endeavours.

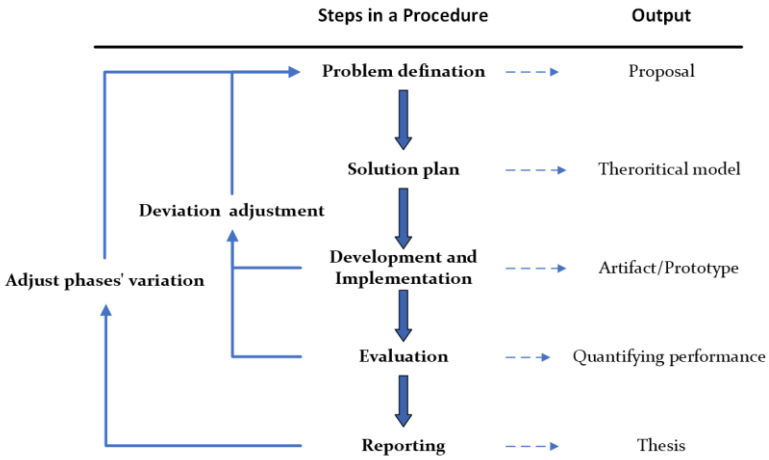


Figure 1.4: Fundamental Stages for Research Methodology.

2. **Solution Plan:** This phase immediately follows the problem definition phase, during which a provisional design is proposed. The preliminary design describes the prospective artifacts and how they can be created. Suggestion is a creative process that demonstrates new concepts, models, and functions of artifacts. The research proposal typically includes the tentative design that results from this stage.
3. **Development and Implementation:** This phase examines the implementation of the tentatively suggested design artifacts. The implementation techniques will be based on the constructed object. The implementation need not be novel; the novelty lies predominantly in the design, not the construction of the artifact. Typically, the development process is iterative, in which an initial prototype is constructed and then modified as the researcher obtains a deeper understanding of research problems. Thus, the output of the suggestion phase is also the feedback of the first step, allowing for the revision of the research proposal. This step includes the sub-steps a) planning, b) analysis, c) design, d) development, e) testing, f) implementation, and g) maintenance to construct the prototype (Niu, Lu et al. 2009).

4. **Evaluation:** This phase involves assessing the implemented artifacts. The efficacy of the artifacts can be evaluated using criteria outlined in the research proposal and the proposed design. The evaluation results, which may fall short of expectations, are fed back to the first two phases. Consequently, the proposal and plan could be revised, and the products could be enhanced.
5. **Reporting:** This is the reporting phase of the design investigation. Typically, it results from contentment with the evaluation outcomes of the developed artifacts. However, behavioural differences exist between the proposed solution and the actual artifacts created. As long as the created artifacts are deemed "good enough," the design research endeavour concludes, and the anomalous behaviour may be the subject of further study.

Due to the case for this research, Figure 1.5 shows the actual implementation of research methodology with our work; it presents the interference of steps of phases one and two. The following chapters seek to answer each research question individually. Moreover, we will discuss the significant common points in offloading and service placement techniques.

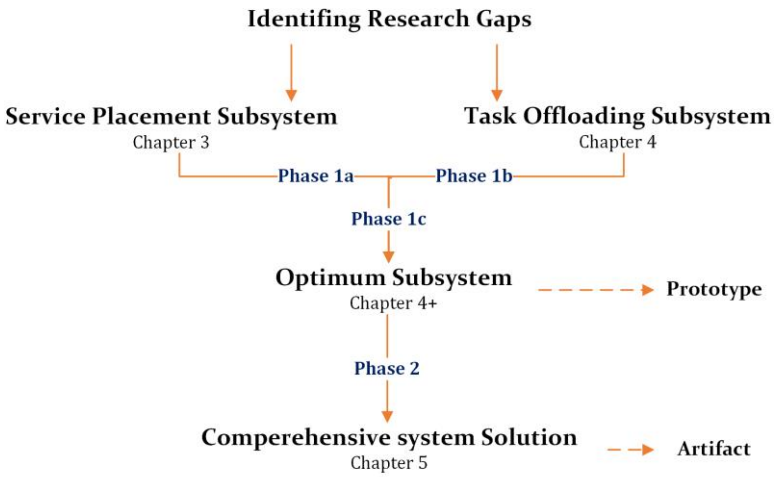


Figure 1.5: The actual implementation of this research.

1.3.2. RESEARCH PLAN

The research plan for this investigation consisted of the following stages:

Step 1. Select a topic: The selection of a research topic can be motivated by observation, personal interest, pieces of literature describing previous theory and research in the field, social concern, or as a result of some current hot topics. This research topic was selected based on previous literature,

research, and the author's observation and experience in the process industry.

Step 2. Literature review: Regardless of the reason for selecting a particular topic, a literature review of previous research in the topic area is a crucial step in the research process. Existing literature in the relevant fields was retrieved and evaluated critically.

Step 3. Finalise research problems: The findings of the literature review contributed to the formulation of the specific research questions for this study. This research study explicitly addressed the research questions. As the clarity and specificity of the research questions increased, more literature closely related to them was reviewed. Existing work is compared to the desired standards; voids and restrictions are identified.

Step 4. Formulate the problem of research: Clients at the IoT layer generate tasks to be executed at fog nodes. When fog nodes get a job, they may send it to the cloud nodes. Because more power will be rented from the cloud because of this process, overall network operating costs will be increased. To solve this problem, an offloading procedure is carried out, which causes more work to be done on the second layer and less in the cloud.

The primary motivation for this work is to override the obstacles that face prevalent offloading to balance the load in the fog layer. Although offloading is the primary motivation, the service placement LB approach will be examined. However, testing more than one technique is appreciated in a PhD project. The innovative offloading technique is presented in this section to explore the main idea behind this work. Both service placement and offloading use the following techniques as possible.

Step 5. Prioritise Sensitive Applications: Several tasks with varying sizes and levels of importance are created at the IoT layer. Although some tasks are pretty large but have a low priority, others are so small that they must be completed on time. Consequently, in the context of our investigation, categorising diverse tasks is essential. This subject will be discussed further in the methods section.

Some of the previous studies have based their criteria for task classification on the assumption that all streamed workloads have the same priorities (Verma, Yadav et al. 2016, Neto, Callou et al. 2017, Téllez, Jimeno et al. 2018, Sarma, Kumar et al. 2019), while a few authors categorise the workloads based on their time sensitivity (Verma, Bhardwaj et al. 2016, Refaat and Mead 2019). Due to the logical conception of service, which uses "first in, first out," the low-priority tasks might be served first, which will obstruct the critical medical case from receiving help. This approach, as seen in Figure 1.6, will affect emergency cases that require immediate action.

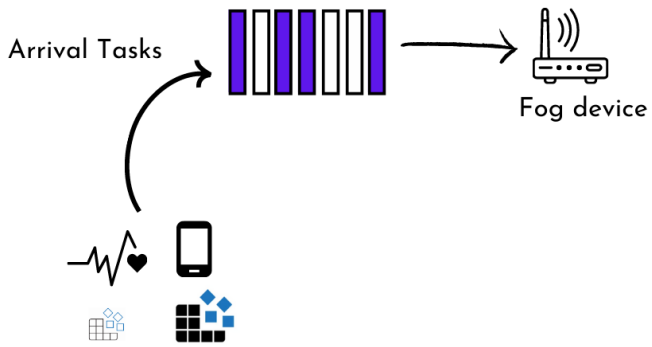


Figure 1.6: FIFO behavior in the proposed model.

The benefit of the second standard method is that it will give credit to sensitive tasks to reduce the waiting time in the queue, which is high due to the high traffic of functions in some regions. Task classification was designed according to the procedure used by (Chen, Huang et al. 2021), which proposes two classes of workloads: light tasks (LT) and heavy tasks (HT). This approach will give emergency cases a chance to execute first, which will improve system latency. On the other hand, heavy tasks will not affect this process due to their requirements. In this case study, we propose a medical computing system with different operating systems that share networks, such as X-ray, reporting, filing, and control systems, which we consider in our case in HT Figure 1.7. When there is heavy function traffic in a particular area, waiting times in the line can be extended. The second conventional approach can help with this problem by crediting sensitive jobs.

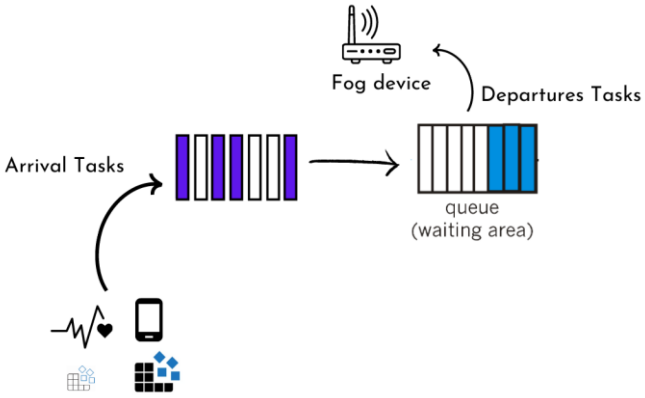


Figure 1.7: Dealing with the Light Task in the Queue.

On the other hand, a system is in place to monitor a patient's vital signs and send an alert in an emergency. LT jobs are given higher priority than other types of tasks in terms of data quantity. As a result, each node incorporates a task scheduler module to reprioritise "LT first" items in the queue. As can be seen in Figure 1.6, this means that even if an HT job is at the front of the queue, we still ensure that it will not be completed before an LT task.

Step 6. Smart Gateway (SG): This proposal employs a straightforward scenario with a cluster of computer nodes to explore LB in this context (see Figure 1.8). A special algorithm is needed to cluster various fog nodes together to produce a clustering of computing power. For example, the method's end goal in the offloading technique is to set up a paired set of compute nodes for direct job offloading without any intermediate node selection.

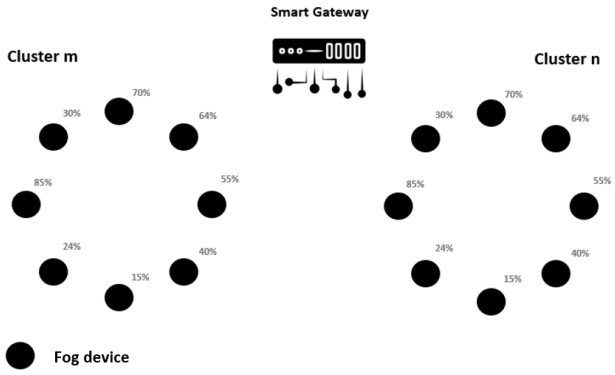


Figure 1.8: Clustering in the proposed algorithm.

The planned system uses fog nodes in critical applications to connect many devices to the network's core. The cloud infrastructure is a vital part of this setup, as is long-term storage backup to help the fog nodes already in place when demand is high. On the other hand, an SG will connect all the FC nodes

and act as a bridge between the local infrastructure and the cloud. However, both service placement and offloading techniques define SG according to their structures. Hence, all communications in the fog layer will be subject to the same suppression level, and the information flow between the fog and cloud layers will be under tight control.

Step 7. Clustering: In this phase, all computing nodes will split into virtual cells to build collective computing power. The SG will use the K-means algorithm to construct the clusters in a different area. Indeed, adjacent nodes in the same geography are clustered into an individual cell, which can have a different size. This philosophy aims to reduce the bandwidth usage of the entire network and limit it to a specific region. Euclidean distance will help to perform distance calculation in the intelligent gateway among nodes in the cluster (Danielsson 1980). Once the smart gateway has all the nodes' positions, the K-mean algorithm, which facilitates clustering in large-scale systems, starts creating clusters. The number of clusters will be decided later.

Step 8. Creating Central Distributed System: The central system can begin static offloading by registering all nodes inside a central dynamic table. The intelligent gateway will regularly broadcast a pair of compute nodes. By taking this measure, we can rest assured that all current nodes will have their load data updated.

Step 9. Evaluation of the proposed algorithm: We will compare the hybrid method against a selective offloading approach to see how well they stack up. The parameters of interest for this analysis are latency times and quality of service. The modelling tool, iFogSim, is ideal for accomplishing this goal.

This thesis strives to achieve two primary goals: fog and cloud computing. The first step in attaining LB in FC's second layer is researching and proposing offloading strategies. First, the newly suggested techniques for LB in fog/cloud computing were investigated using an extensive study. All viable offloading systems have been uncovered through this investigation. After this phase, we will have a reinforcement offloading approach that considers the benefits of existing models while minimising their downsides. To achieve this goal, we will investigate the shortcomings of existing offloading methods and

develop an effective model to maximise the efficiency of the already available processing nodes.

Step 10. Sustainability: Even though the offloading technique would improve the LB level, it would cost the entire system's resources. Most offloading algorithms follow a dynamic strategy to unload rather than a static one, which leads to accumulating offloading time. Thus, finding a creative technique to avoid this accumulated time is vital. Moreover, a massive number of messages exchanged among nodes will cause high pressure on the infrastructure of the network's backbone.

The second aim of this thesis is to create a sustainable solution for LB without causing another issue. Figure 1.9 shows the main framework of the proposed model.

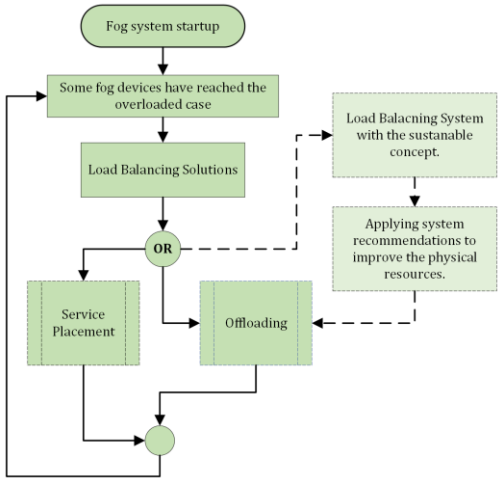


Figure 1.9: The Proposed solution for the research problem.

1.4. Research Frameworks

The problem to be addressed through this study is the increasing latency in time-sensitive applications due to the unfair resource allocation of the arrival workload among the computing units in the fog layer, especially in large-scale networks. Creating a balanced computing environment in the fog layer decreases the latency, enhances performance in sensitive applications, and utilises the available resources in this layer.

1.4.1. Theoretical Framework

Building a comprehensive framework requires a well-known understanding of all the research concepts. Many concepts in research must be highlighted here. This

subsection lists and discusses the concepts used during the literature review stage, where these concepts are repeated in most of the prior studies. Figure 1.10 depicts the rapport between them.

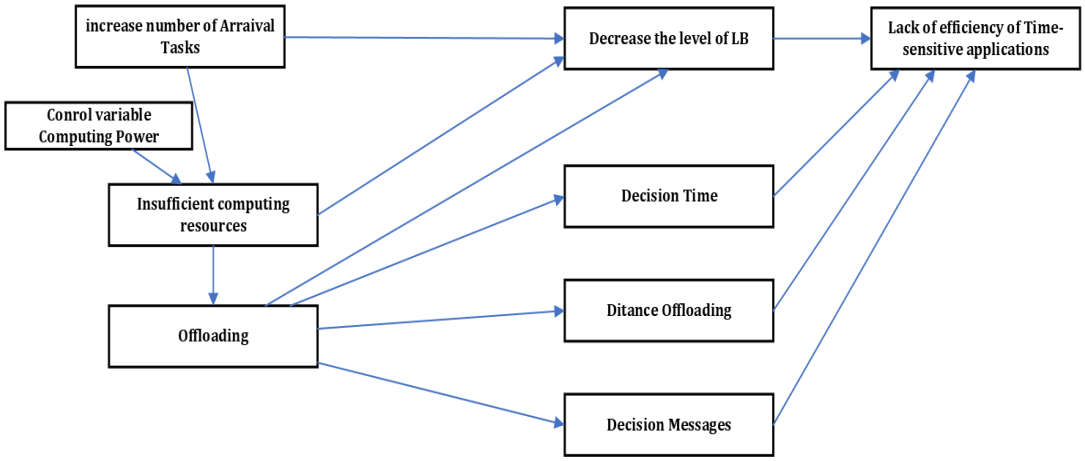


Figure 1.10: The rapport among variables of research.

- **Computing load:** There are many functions for computing devices, such as storage, computing, and networking. Computing is the core function of any smart device. In FC, the Fog device is the central computing unit in the layer. The fog device works by executing the arrival task from the IoT layer. However, most fog devices have a limited computing capability, forcing the served device to list the arrival task in a private queue. This action generates a variable time, named a waiting time. Waiting time is the time computing units consume to execute the queued task. It relies on the computing power of the fog device and the rate of arrival tasks. However, this work defines a *computing load* as a dependent variable because these factors affect it. As the computing power cannot be changed, we consider the arrival task as the independent variable affecting the computing load, where computing power is a control variable.
- **Offloading:** There are many reasons for offloading to be used in computing fields. This technique shifts the workload among computing units for different reasons. It is used primarily in multi-processing systems (Albalawi, Alkayal et al. 2022). In this study, we utilise offloading to migrate the arrival tasks in loaded computing units to the underloaded units. To make the offloading decision, we need to create an algorithm to find the optimum device for offloading. The target device will return the computing results

after accomplishing the task. Aazam, Zeadally et al. (2018) argue that offloading operations enhance the LB level among fog devices. Consequently, the offloading process initiates if the fog device faces a high load. Accordingly, we consider offloading as a dependent variable.

- **Arrival Tasks:** The Arrival Tasks or workloads are the overall amount of work assigned to someone. Firstly, we need to define the task in FC. The study describes it as the request sent by an object in the IoT layer to be executed. Regardless of the size of the computing power of the IoT objects, many cases are required to send the data to be processed in fog or cloud layer due to the availability of on-demand service. However, we call the tasks that arrived in fog or cloud layers "arrival tasks".
- Consequently, we can define workload as the number of arrival tasks sent by IoT objects to higher servers to be executed. As a requirement in a theoretical framework, we need to identify the type of workload. Hence, the workload is an independent variable because they are generated outside our framework.
- **Computing Power:** The amount of computing resources that can be used to complete a specified number of tasks. It can be measured by the processing capacity and speed of individual components, such as processors and memory modules, or by the total computing resources of an entire system, such as a data centre. Typically, the computational capacity of a system is determined by the resources accessible to its users. In this research, we consider the computing devices as homogeneous units. Therefore, it is a control variable.
- **Computing Resources:** Several computational resource-related topics may require your consideration as you plan your research. Following are descriptions of the most prevalent resources. The purpose of this document is to assist in identifying resource requirements and to initiate a discussion on how to meet those requirements. This research identifies CPU power as the vital resource that needs to be secured to accomplish task execution. The capacity of the CPU limits it.

1.4.2. CONCEPTUAL FRAMEWORK

This section presents the conceptual framework for this thesis. It consists of seven steps that must be performed to generate a sustainable LB system in a fog environment. During the PhD research, we performed these steps.

In the first year of the research, selecting the research gap was the main issue after choosing FC as the primary research area. Many topics were found, and most of them were interesting. However, LB is selected after reading and checking many review papers in this domain. Next, and in the solution plan, we identify two methods to balance the load in FC. Service placement is the first technique to balance the load, while task offloading is the second technique. Later, as part of our research method, we selected iFogSim as a simulation tool to build our proposed model. In the data analysis stage, the outcomes of the experiments are reported and documented, and the third year focuses on accomplishing the remaining steps, as shown in Figure 1.11.

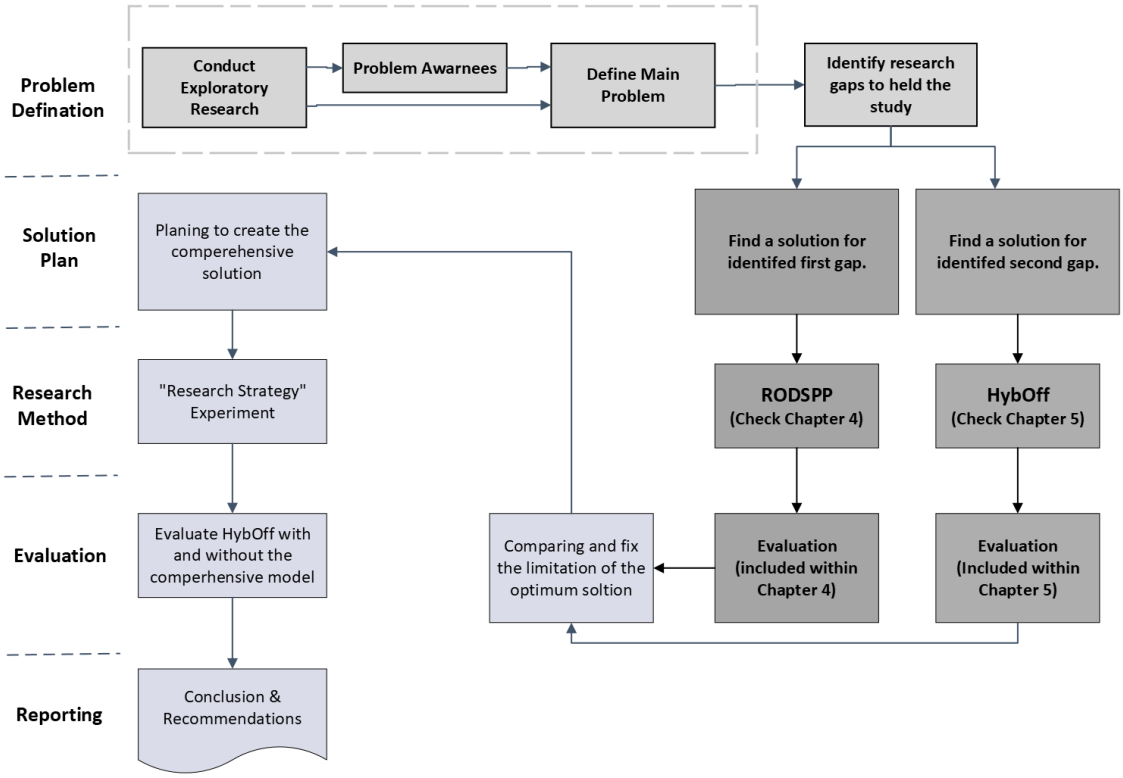


Figure 1.11: THE CONCEPTUAL FRAMEWORK OF THE STUDY.

1.5. THESIS STRUCTURE

This is the general structure of the thesis, as shown in Figure 1.12.

Chapter Two: The state-of-the-art FC computation offloading, and LB are reviewed and discussed. This chapter follows the systematic review to present the topic.

Chapter Three: This chapter attempts to balance the load of the fog layer by enhancing the service placement policy. We propose reinforcement optimisation for a decentralised service placement policy (RODSPP), which attempts to mitigate some of the drawbacks of existing placement policies. This chapter aims to answer the first research question.

Chapter Four introduces the second attempt to improve the LB in FC by proposing a novel offloading model. This chapter attempts to answer the second and third research questions titled "How to create a practical offloading model that improves the efficiency of existing FC nodes as far as possible?" and "How to determine the optimal technique among the two techniques?" respectively. To our knowledge, this is the first work to introduce hybrid offloading in the real meaning.

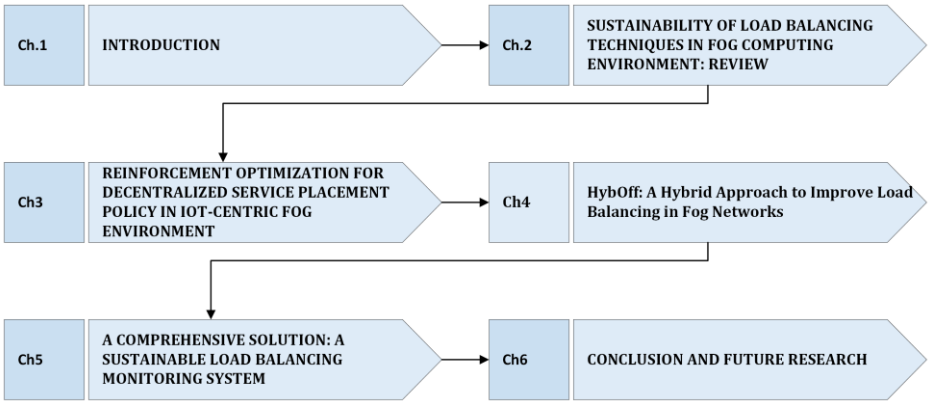


Figure 1.12: Thesis Organization.

Chapter Five: In this chapter, we attempt to answer the fourth research question: "How to create a sustainable solution for LB to avoid overload situations in FC?". It presents the second and last phase of the thesis research. It proposes a SlbmS that seeks a long-term concept using machine learning.

Chapter Six: This part of the thesis concludes the research with a discussion of limitations and potential future research.

1.6. PUBLICATION RELATED TO THIS RESEARCH

In the Academic field, the publication is vital. PhD is one of the academic degrees that need to experience the ability of student ability in publication. "More publications, more respective in this field" is the standard clause in the academic field. Regardless, we have designed the research framework to have the ability to publish its phases. Table 1.1 presents the updated states of each paper per phase.

Table 1.1: THE PLANNING OF THESIS PUBLICATIONS.

Sr.	**Article**	**Phase**	**States**
1.	*Sustainability of Load Balancing Techniques in Fog Computing Environment, Review.*	Pre-phase I.	Published
2.	*Reinforcement optimisation for decentralised service placement policy in IoT-centric fog environment.*	Phase I- Stage 1.	Published
3.	***HybOff**: A Hybrid Offloading Strategy to Improve the Performance of Time-Sensitive Applications for Large-Scale Networks in Fog Environments.*	Phase I- Stage 2.	Peer Review-Publish stage
4.	*A Comprehensive Solution: A Sustainable Load Balancing Monitoring System.*	Phase II.	Peer Review
5.	*Sustainability of Load Balancing Techniques in Fog Computing Environment, Systematic Review.*	General.	Draft

2. CHAPTER, LITERATURE REVIEW

Abstract: To increase the number of real-time and sensitive-time applications, reinforcing and supporting edge computing, such as FC, is required. FC remains a fertile field for research, even with the advent of other computing systems. On the other hand, a tremendous number of Internet of Things (IoT) objects and their requests make FC face obstacles to providing the required Quality of Service (QoS). The distributed nature and limited resources of fog systems make cooperation among their devices challenging to handle user requests. LB is one of the practical techniques that can remedy this situation. Many researchers seek to introduce innovative mechanisms to create balanced computing devices. Utilising the sustainability concept to solve the dilemma of LB in FC is the area of this research. **However,** this chapter presents an empirical study of a taxonomy of existing techniques in LB. Besides, it summarises the research study according to the approved review protocol, which defines the extracted information from each article. The chapter finds the best practices in state-of-the-art research to answer research questions; this answering paved the way to solving the research problem with a sustainable concept.

2.1. INTRODUCTION

In recent years, the cloud computing model has acquired popularity. This computing architecture has garnered widespread adoption in the IoT as an effective means of managing and centrally processing data. The two primary components of cloud computing are the cloud and the Internet of Things, or IoT, client layer. Numerous IoT client-layer devices and objects incorporate sensors for gathering various environmental measurements. The IoT layer processes and transmits the information to the cloud layer. Over time, the cloud system collects and stores a mountain of data for analysis from these devices. However, increasing the reaction time could result in the abandonment of cloud technology for specific applications, such as intelligent infrastructure, augmented reality, virtual reality, and healthcare. (Aazam, Zeadally et al. 2018).

Later, FC was introduced as a new technology to bridge sensitive cloud and IoT technology gaps, such as time latency, location awareness, and service quality. FC complements the cloud environment by occupying a layer between the client and cloud layers. Although the cloud layer has allotted heavy computational burdens to support lower layers, the fog layer has been designed to alleviate the cloud load by performing tasks at the periphery. In contrast, portions of fog devices, such as the central server in a high-density area, become overwhelmed due to the vast amount of IoT-generated data in the client layer. Moreover, FC's utilisation has increased significantly due to the accelerated growth in the usage and implementation of IoT applications (Mahmud, Kotagiri et al. 2018). These overburdened fog nodes have laboured due to the high demand for services, which has led to unstable computing environments. As a result, some nodes are heavily occupied while others are inactive. Intelligent techniques must redistribute the transmitted duties among the computational devices in all layers to create a balanced computing environment and to increase the demands on overburdened nodes to address this issue.

Alternatively, the World Commission on Environment and Development (WCED) introduced the term "sustainability" in a broad context during the 1984 United Nations General Conference. Nevertheless, many researchers define sustainability from a computing standpoint; they introduce it using the same concept. González-Mejía, Eason et al. (2014) described sustainability as

"maintaining a system's functionality without experiencing significant deterioration over time." Many experts have studied sustainability in various computing contexts to prevent the waste of available resources and maximise their utilisation. However, many papers have proposed solutions to improve fog and cloud computing performance without considering sustainability.

LB, on the other hand, is an effective method for enhancing the entire computing environment and ensuring that all computing components operate simultaneously. Implementing the most recent LB concepts to any computational system will improve the complete system's performance, and all nodes will be burdened equally. However, many researchers have recently conducted studies in LB and FC. Numerous papers have proposed and introduced a competitive computing system at the network's periphery since 2015. Nevertheless, the LB techniques present innumerable obstacles that may diminish the efficacy of such a solution. Efficiency, overhead, implementation consequences, and improving defect tolerance are some of LB's current challenges.

This literature review investigates recent articles and knowledge on LB in the fog domain by addressing the following points: a comprehensive primer to FC, including an overview, definition, principal components, typical architecture, and application examples. It illustrates LB's core definitions, concepts, scenarios, proposed solutions, prose, disadvantages, and controller categories. Are there any proposed LB remedies that incorporate sustainability to address this issue? It investigates and discusses recent offloading algorithms and their rapport with the concept of sustainability, if any. The overall structure of the research is comprised of two components, which are the thesis's most significant contributions.

The remainder of this chapter is organised as follows: section 2 presents and discusses the related systematic research, while Section 3 presents the review methodology used in this chapter. Sections 4, 5, and 6 present the background of the study, followed by the challenges faced by the FC in section 7. The chapter output is discussed in section 8. Section 9 introduces the conclusions.

2.2. Related Review Studies.

Although many review papers have been found at the start of this survey, they are not recommended to be considered when systematic review papers are available (Brereton, Kitchenham et al. 2007). Three systematic reviews have been found on LB solutions in FC (Kaur and Aron 2021, Kashani and Mahdipour 2022, Shakeel and Alam 2022). However, to ensure the necessity of this review, the existing systematic reviews should be investigated as suggested in (Kitchenham 2004).

In a fog environment, Kaur and Aron (2021) presented the first systematic review paper on LB. The authors covered 72 research papers published between 2010 and 2020 using specific databases like IEEE Xplore, Science Direct, Springer, Google Scholar, ACM digital library, and Elsevier. They studied the selective articles from different aspects, such as their algorithm, pros, cons, and simulation tool used. Based on that, they introduced a taxonomy for these solutions and potential research gaps. The study categorised the existing LB algorithms into two main types: static and dynamic. Though the authors classified all static algorithms under one class, they classified the existing dynamic LB into five categories: traditional, agent-based, hybrid, real-time, and nature-inspired. Moreover, they proposed their architecture and solution for LB in a fog environment. However, the authors answered the research questions essential to understanding the fog system clearly.

After that, and in the second review paper, Shakeel and Alam (2022) discussed different sides of cloud and FC. The article investigates and studies around sixty papers published from 2015 to 2022 using unclear databases. However, they declare some of them, such as IEEE Xplore, Springer, Elsevier, Wiley, and IGI Global; Table 2.1 shows the differences among the used databases. It also presents different taxonomies for LB algorithms. The authors followed the main classification (Kaur and Aron 2021). However, the subclassifications were different, i.e. sub-optimal and optimal are the subclassifications for static LB, whereas control and component-based are the subclassifications for dynamic LB. However, the paper does not include any research questions, which is a piece of crucial knowledge for any systematic review paper.

Table 2.1: THE USED DATABASES.

Name	Type	Disciplines
ACM Digital Library	Electronic database	computer science (informatics)
IEEE Xplore Digital Library	Electronic database	Computer science, electrical engineering, and electronics
Elsevier	Dutch academic publishing	humanities, and the scientific
Wiley	USA academic publishing	Science, technology, medicine, professional development, higher education
Science Direct	Indexing system	scientific and medical publications
Taylor & Francis	England academic publishing	Behavioural Science, Education, Law, Science, Technology, Engineering, and Mathematics, Medicine
Web of Science	Indexing system	the sciences, social sciences, the arts, and humanities,
IOS Press	Netherlands academic publishing	Scientific, technical, and medical research
ProQuest	Indexing system	Information and data provide

Kashani and Mahdipour (2022) wrote the third systematic review of LB mechanisms in FC. The authors covered several search engines like IEEE Xplore, Science Direct, Springer, ACM digital library, Taylor & Francis, and Wiley. The study selected 49 articles published between 2013 to 2021. The study introduced most of the classifications, metrics, and simulation tools for LB raised by research questions. However, the authors adopted different classifications for LB; approximate, exact, fundamental, and hybrid are the main classifications of LB in this study. The adopted classification in this article was entirely different from the previous papers. Table 2.2 depicts the differences between the systematic review studies based on surveys (Kaur and Aron 2021, Kashani and Mahdipour 2022, Shakeel and Alam 2022) and this research.

Table 2.2: A COMPARISON AMONG DIFFERENT SYSTEMATIC STUDIES OF LB APPROACHES IN THE FC.

Ref.	Scope of Research	Pub. Year	Reviewed articles	Taxonomy	RQs	Covered years
(Kaur and Aron 2021)	General	2021	72	Approximate, Exact, Fundamental, and Hybrid	5	2010-2020
(Kashani and Mahdipour 2022)	General	2022	49	Static and Dynamic	6	2013-2021
(Shakeel and Alam 2022)	General	2022	60	Static and Dynamic	N/A	2015-2022
Our Survey	Focusing on time-sensitive application	2023	103	Static, Dynamic, and Hybrid	5	2015-2023

2.2.1. THE MOTIVATIONS FOR THE SYSTEMATIC REVIEW

The primary motivation for the research is to study the overlap area among FC, LB, and sustainability concepts. Many terminologies, concepts, and metrics needed to

be clarified in this area of research. This review defines and summarises most of LB's terminologies, concepts, and taxonomy in FC. It is considered a decent start for new researchers interested in this field. Moreover, this review will draw more general conclusions about FC and LB techniques based on existing systematic reviews. On the other hand, this review discusses the sustainability concept in the computing field. The review protocol is the key to the planning of this study, as defined before.

Consequently, we found that our review will be different because it focuses on specific research areas. This review will synthesise the taxonomy of the existing studies to create a comprehensive one. It also provides some approved calculations for some metrics. Thus, this article has value when it is written as planned.

2.2.2. Research Contribution

A list of points makes this review unique over other existing systematic reviews for this research area, as shown next.

- This chapter is the first systematic review in which writing relies upon three existing systematic reviews.
- It focuses on the offloading techniques for time-sensitive applications in a fog environment with the support of the sustainability concept.
- It introduces some selective calculation formulas to each metric.
- It explores the sustainability concept in the existing solutions (if any).
- Furthermore, it provides a different taxonomy than the existing SLRs.

2.3. Research Method

As a requirement for a PhD degree, we summarised the existing research about the strategy of LB in FC. The summary follows the systematic review described in (Kitchenham 2004) thoroughly and unbiasedly. However, Kitchenham (2004) argues that any new systematic review must identify and review the existing systematic reviews of the phenomena of interest. Thus, we followed the checklist Khan, Ter Riet et al. (2001) suggested to evaluate the current systematic research, as discussed in section 2.2.

However, planning, conduction, and documentation are three-stage procedures obtained from (Brereton, Kitchenham et al. 2007) and the guidelines in (Kitchenham 2004). To make this work fairer, we consult an external assessment to verify the results of each research step; moreover, the external evaluation engages in a precise

classification of the existing algorithms. The three stages of the methodology are abstracted, as shown in Figure 2.1, to perform the target of this chapter efficiently.

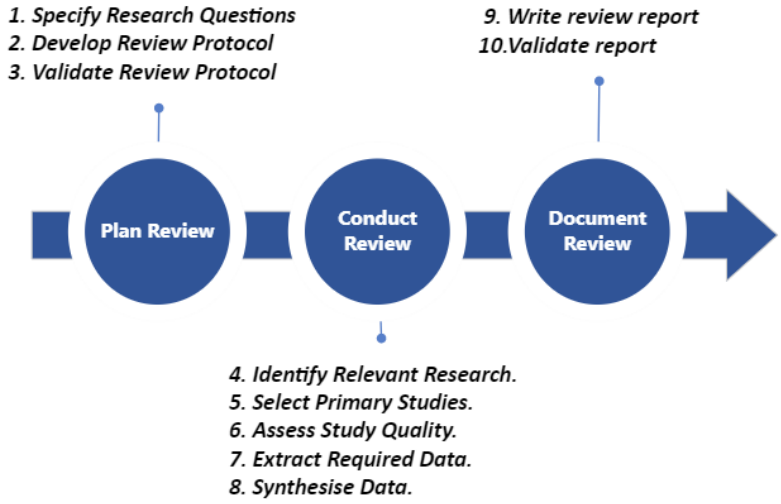


Figure 2.1: The stages of the Research Methodology.

2.3.1. Planning Review

The planning stage is the first step towards building a comprehensive systematic review. It contains the basic rules which govern the following stages. Writing flexible and valid rules facilitates the conduct and document stages.

2.3.1.1. Specifying the Research Questions

To enrich this review chapter, we need to answer the research questions of this thesis, which was one of the main motivations of this study. This research seeks to answer nine questions crucial to understanding the research problem and the existing solution (if any). The aim of investigating each question is outlined by motivation. On the other hand, a comparative analysis enables an examination of the collective impact of the research, which is presented in terms of comparison characteristics. During the research review, the research questions (RQs) are considered.

2.3.1.2. Developing the Review Protocol

In contrast to unstructured review approaches, systematic reviews reduce bias (Brereton, Kitchenham et al. 2007) and follow a precise sequence of methodological to research literature. Kitchenham (2004), “A systematic review relies on validated and a truly-defined review protocol for extracting, analysing, and documenting the results”. It attempts to avoid the researcher’s expectations in the selection process

of the individual studies. This systematic review follows the subsequent review protocol guidelines, such as:

- Background: The article seeks to investigate most of the proposed solutions for time-sensitive applications that have been published in ranked journals in the area of research.
- The article must seek to answer the Research Questions (RQs) mentioned in Chapter One. Therefore, an initial scoping study is built to collect the required data.
- After performing the initial scoping study, the approved searching strategy has been defined. The method for searching primary studies includes search terms and a list of databases. The search terms are "Load balancing" + "Offloading" + "Resource allocation" + "Service placement" + "Fog computing". Regarding the searched database, we have chosen IEEE Xplore, IOS Press, Springer, Web of Science, ProQuest, Scopus, and Wiley.
- *Study Selection criteria* are the conditions that must be satisfied by each selected article. It consists of including and excluding points.
- The review protocol must contain developed *quality checklists* to evaluate the individual studies.
- Data extraction strategy. The extraction data for each selected article contains metrics equations, methodology, results, challenges, simulation tools, and compared algorithms. Moreover, we need to check if the reviewed article answers any research questions of this study.

2.3.1.3. VALIDATING THE REVIEW PROTOCOL

Based on the purposes of this article, the RQs and the study scope were specified to make the search strings for extracting literature. The review protocol was also developed for a systematic study by following (Kitchenham 2004) and the experience with existing systematic reviews (Kaur and Aron 2021, Kashani and Mahdipour 2022, Shakeel and Alam 2022). Nevertheless, as Kitchenham (2004) has recommended, the review protocol has been reviewed and criticised by the supervisor, who is considered here as the external assessment.

Before the execution process for the review protocol, an external assessment, which had experienced a systematic survey process before, was performed and

modified depending on its feedback. To build the *initial scoping study,* we applied the approved reviewed protocol to ten articles to make the initial scoping study. The output of this stage prevents us from struggling among the high number of searched articles. During the experimental studies, the study scope was expanded, the search methods were enhanced, and the exclusion/inclusion criteria were refined Table 2.3.

Table 2.3: THE EXCLUSION AND INCLUSION CRITERIA.

Inclusion	• Research articles that introduce innovative solutions or algorithms on LB in FC focus on time-sensitive applications. • Peer-reviewed articles in conferences or journals • Articles published between 2015 and August 2023
Exclusion	• Articles that study Service Placement, Resource Allocation, task scheduling, task allocation, IioT, vehicular or IoV systems, • Review articles, editorial articles, short articles (less than six pages), write articles, and non-English articles. • Research articles that do not mention solutions and algorithms to improve LB in FC explicitly. • Books, book chapters, and dissertations

2.3.2. CONDUCTING AND DOCUMENTING THE REVIEW

After we finalise planning the review, we will conduct the review. The conduction stage consists of five steps. Identifying the relevant research and selecting the primary studies are the first two steps, followed by assessing the quality of the selected papers. Then, Extracting the required data and synthesising them are the last two steps in this stage.

2.3.2.1. RELEVANT RESEARCH AND PRIMARY STUDIES

The conduction stage consists of four consecutive steps. However, this is the first stage, which aims to select the primary studies. This subsection lists the procedures for searching and selecting articles in the second phase of this review. We need to identify relevant research as described next to accomplish the selecting step.

We have identified relevant research. The search string or keywords that follow are used within the different databases to find the articles related to our study. This process searched abstracts, titles, and keywords in Well-known electronic academic databases like IEEE Xplore, IOS Press, Springer, Web of Science, ProQuest, Scopus, and Wiley are employed. The output of this step was 582 articles, as shown in Table 2.4. Notice that the search process was between 2015 to 2023. Even though FC was

introduced in 2012 (Sulimani, Alghamdi et al. 2021), we have chosen the starting year 2015 to find articles more robust.

Ultimate selection- by applying the inclusion and exclusion criteria, we examined the extracted 308 articles. Then, the articles that had passed the quality assessment were chosen. By the end of step three, 103 studies were finally selected to be included in this review.

Table 2.4: SEARCHING PROCESS.

Serial	Indexing & Database Systems	Outcomes		
		Step I	Step II	Step III
1	IEEE Xplore	64	44	32
2	IOS Press	35	8	4
3	Springer	213	83	16
4	Web Of Science	114	65	19
5	ProQuest	15	11	10
6	Scopus	134	94	19
7	Wiley	7	3	3
Total		582	308	103

2.3.2.2. DATA EXTRACTION AND SYNTHESIS

As mentioned in the section on review protocol, a structured format was designed as described in (Kitchenham 2004) to record the target data from selected articles. An organised comparative analysis was performed by analysing each article, enriching the article with an investigation of the collective research impact.

2.4. BACKGROUND OF RESEARCH

In 2012, Cisco's researchers proposed FC as a new term in the computing world (Yannuzzi, Irons-Mclean et al. 2017). Processing data at the edge was not a novel concept then, as edge computing and cloudlets had emerged before 2001 and 2009. Both FC and Cloudlets are the revolution of a similar notion, which revolves around computing and analysis at the border area. FC is available to connect the IoT's objects. It is a newly emerging decentralised computing paradigm that closely connects with central cloud computing and IoT devices. On the ground, the extreme data generated via IoT devices such as sensors, cameras, and wearable devices can be analysed (processed) locally in fog devices, e.g., routers.

In contrast, the cloud-based systems are forwarded to the cloud in the network core (Mouradian, Naboulsi et al. 2017). The enhancement of FC frameworks produced the administrators' choices for computing data at the most suitable location in the network to increase the QoS. For example, many applications might

require high reliability and ultra-low latency processing tasks, such as in a connected vehicle use case where vehicles are required to respond to an accident in less than 2 milliseconds of delay (Stojmenovic and Wen 2014).

FC offers several benefits for IoT applications, such as performing all steps of data transmission, data extraction, data execution, and data storage from the IoT apps (Xu, Fu et al. 2018). It can fulfil such high-reliability and low-latency service requirements and boost the processing capacity for the network by utilising the excellent performance of the network connections among the fog devices at the edge and the analytics Edge points (Naha, Garg et al. 2018). Also, FC can minimise the required bandwidth of the background links), compared to cloud computing, where the whole data set needs to be forwarded to a data centre in the core network for processing and storage (if necessary). The future IoT networks and era will adjust to this scenario, where a noticeable volume of data can flow into the system for computing, storage, or analysis. In other words, FC is very similar to obsolete computing systems, where the processing occurs near the devices or locally without sending them to the remote systems, yet with a central system with powerful processing and communication facilities.

2.4.1. FOG COMPUTING (FC)

The interest in and focus on FC research is rapidly increasing. Several FC definitions rely on different points of view. One of the first people to define FC was Bonomi, Milito et al. (2012), who wrote: "*Fog computing is a highly virtualised platform that provides compute, storage, and networking services between IoT devices and traditional cloud computing data centres, typically, but not exclusively located at the edge of the network*.". This definition is superficial; it does not consider the fundamental aspects of the spread nature of FC; it is just an upgrade in organising abilities, like by providing facilitating conditions and improved help for communication between gadgets.

Vaquero and Rodero-Merino (2014) defined FC as "*a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralised devices communicate and potentially cooperate with the network to perform storage and processing tasks without the intervention of third parties. These tasks can be for supporting basic network functions or new services and applications*

that run in a sandboxed environment. Users who lease part of their devices to host these services get incentives". Although this definition has explained the fog system from different angles, it ignores the relationship between cloud computing and fog environments and why we need to use this type of technology.

Another researcher intends to define this technology as *"Fog computing is a distributed computing platform where most of the processing will be done by the virtualised and non-virtualised end or edge devices."* (Patwary, Naha et al. 2021). Moreover, Naha, Garg et al. (2018) *define "It is also associated with the cloud for non-latency-aware processing and long-term storage of useful data by residing between users and the cloud.".*

According to a definition provided by Mukherjee, Shu et al. (2018), FC is a novel model that expands the traditional cloud computing model to the network's edge. In some application components, processing (e.g., latency-sensitive ones) is executed at the network's edge. In contrast, others (e.g., computationally intensive and delay-tolerant components) can occur in the cloud centre. Networking, computing, and storage services are the basic building blocks of the cloud and fog that can provide this service. It is the most comprehensive definition due to enough information about the fog system.

Other researchers introduce the FC as *a decentralised computing paradigm with a pool of resources consisting of more than one ubiquitously connected heterogeneous (wireless or autonomous) device at the network's edge. It might work in a standalone mode without supporting cloud services to provide the clients with available resources collaboratively* (Yi, Qin et al. 2015). Baburao, Pavankumar et al. (2021) define FC as *a distributed computing model that provides all the benefits of cloud computing at the boundaries of the network and acts as a cloud to the clients at the edge*.

Although the expressive definition for FC is *"clouds closer to the ground*" (Mukherjee, Shu et al. 2018), we can modify *that* to be "*fog computing is a mini virtual cloud that resides closer to the ground,"* which will be more descriptive. Even though this definition is too simplistic, it is *expressive*.

2.4.1.1. Architecture of Fog Computing

A new emerging platform needs standard engineering to be deployed. Currently, FC has no approved standard design (Iorga, Feldman et al. 2017). Nevertheless, many researchers have introduced fog processing architectures.

High-Level Architecture of Fog Computing

The FC platform is formed by high-level architecture, categorised into three vertical layers, including Cloud, Fog, and IoT, respectively, as shown in Figure 2.2.

Layer 1: The ground layer is the IoT level, where all connected devices, such as mobiles, tablets, and sensors, exist. In this layer, devices are distributed *geographically* and perform the sensing and actuation processes as an extra activity. For example, while IoT devices nowadays provide processing capabilities, computing power is an additional function for the sensors. The time-sensitive processing should be performed exclusively on the fog layer, while the processing is not time-sensitive and can be performed in the cloud layer. The fog layer decides what should be forwarded to the cloud layer and what should not. In this situation, the clients can gain services from the cloud and the fog based on their condition. However, the complex tasks will be managed by the cloud layer.

Layer 2: The fog layer is the most crucial layer. It is a group of nodes with idle storage and/or computational power. Thus, it might have a uni-computational node or several connected devices communicating dispersedly (Chandak and Ray 2019). This layer contains all intermediate processing and analytic devices, such as switches, gateways, servers, and routers. These devices can perform traditional virtualisation, typically as the cloud; however, this layer accumulates all generated data by the IoT zone and pushes the required data to clients after processing. For example, while routing data is the primary function of a router, storage and computing are extra activities for the router. Even though the big data issue can be fixed by processing accumulated data at the edge zone, billions of connected objects will initiate big data problems. Thus, it employs small- to medium-scale big data computing at this level. Many researchers' work has been established to expand the knowledge of big data on the Fog platform (Oueis, Strinati et al. 2015, Ningning, Chao et al. 2016, Refaat and Mead 2019).

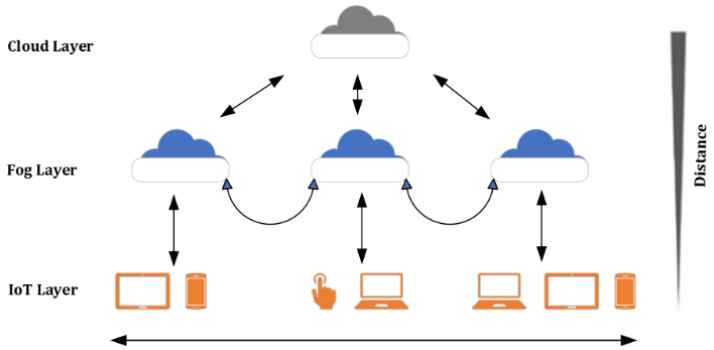


Figure 2.2: Fog Architecture.

Layer 3: The cloud layer is the top layer in the fog architecture, comprising the processing servers and storage servers. These servers have extensive storage computation power. It is an essential part of any IoT-based system that relies on a central location and is accessible worldwide through the internet.

CHARACTERISTICS OF FOG COMPONENTS

Figure 2.2 illustrates a basic model of FC. There are many essential computation components in fog environments, such as fog servers, gateways, routers, and switches. In other words, we can consider any device with storage capabilities, networking capabilities, and computation power to be a fog device. All these devices are managed by fog servers or intelligent gateways, which act as coordinators for services among heterogeneous objects in the client layer. Thus, the fog server is responsible for adjusting the communication among IoT devices, and it is also responsible for establishing and maintaining communication links among the devices' layers (Wu, Liu et al. 2019).

Although both fog and cloud share flexible resources of storage, computing, and networking, it is obvious to highlight the unique characteristic of the FC model, which decides to implement the fog model in place of the cloud paradigm crucial as follows (Mukherjee, Shu et al. 2018, Téllez, Jimeno et al. 2018):

Edged position: The minimum latency requirement for applications such as video streaming, augmented reality, and gaming has required various types of networks. While the fog system is located at the edge and close to IoT devices, it has the power to support latency-sensitive programs that rely on real-time processing. Generally, IoT applications with location awareness on the computing servers would make a difference.

Edge Analytics: regardless of centralised analytics in the cloud platform, FC can analyse sensitive tasks locally. In a cloud scenario, the task must be sent to the cloud (unique computing power in the system) to analyse the data. In this case, the travel time will be added to the total time, increasing the execution time.

Location awareness: Because of the enormous increase in mobile applications in recent years, FC, as a dominant force in network computing, needs to improve the QoS to tackle the explosion of tasks generated by mobile devices. Even though the computational capacities of mobile devices and IoT devices are increasing, the QoS demands of some applications will continue to rise. Besides that, many of those applications will require information on location awareness from the system they are running. Although it has limited computational capabilities, it can derive its location. Thus, the system can track the client's device position.

Scalability or Geographical Distribution: Many cloud-owning companies such as Amazon, Google, and Microsoft have been able to arrange their cloud data centres across the world to improve QoS for their customers and alleviate the load on the primary data centres. For example, the Google Company has distributed cloud data centres in Asia, South and North America, Europe, and Australia (Haris and Khan 2018). Although cloud computing has the advantage of covering a wide geographical area, it often becomes the bottleneck of the whole system. On the other hand, FC can form mini clusters of nodes to be more efficient by being close to the clients.

Wireless access networking: While FC uses Wi-Fi technology to communicate with its nodes, fog nodes can communicate distributively.

Federation and seamless interoperability: FC comprises multiple computing nodes widely distributed and interconnected to cover the workplace. Therefore, the block of nodes in that context is possible and beneficial to tackle the vast number of tasks generated by clients and IoT devices. Moreover, large-scale sensor networks support a heterogeneous end-user and device due to edge devices' proximity to the compute nodes. FC networks are planned to be supported by many sensors.

Low latency: A wide range of applications will directly benefit from applying the fog paradigm to their network by installing computing nodes closer to their sensors or IoT devices. This technique will work to decrease the processing time for the

requests. Vehicular networks are an obvious example that have a clear benefit from fog. Because latency is a crucial vector, FC is mandatory (Hou, Li et al. 2016).

OPENFOG REFERENCE ARCHITECTURE

As the use cases show, an FC framework can have a pool of gateways and components, such as switches, routers, gateways, and other IoT devices. This might require a specific design of the FC paradigm to get the most out of FC. The OpenFog Consortium was made by more than fifty organisations worldwide, led by Cisco. It has set three goals for developing a fog framework. Figure 2.3 illustrates the architecture of OpenFog. The OpenFog research architecture is a mid-level heuristic document full of valuable model suggestions for specialists planning to implement fog networks, fog components, fog-based applications, or fog nodes. It explains several mentions of fog use cases in smart cities, visual security, and transportation.

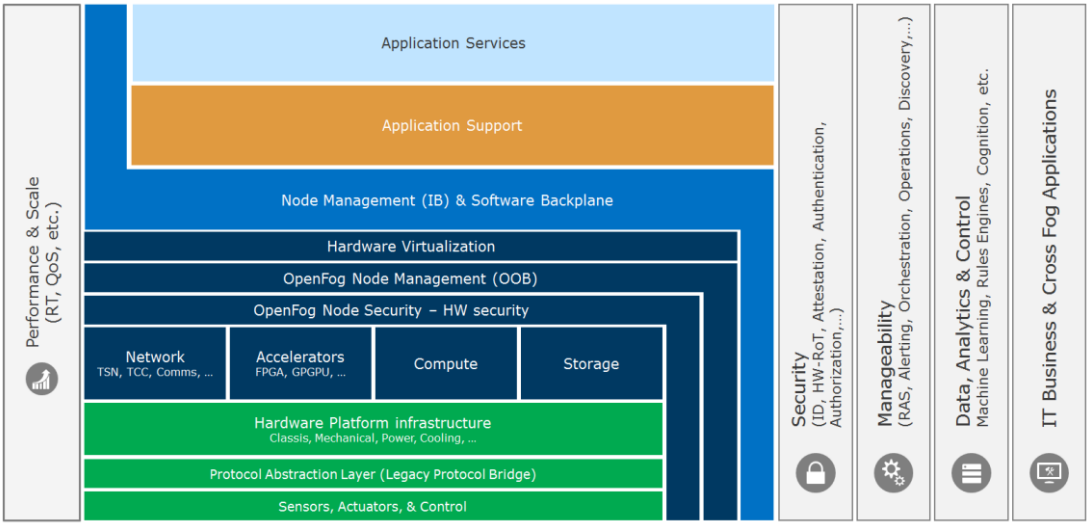


Figure 2.3: OpenFog Architecture.

Eight pillars of OpenFog constitute the core of its capabilities, such as openness, autonomy, security, scalability, agility, program ability, manageability, and hierarchical organisation. The research architecture has discussed these pillars in detail and shows the interrelationships among various components. OpenFog's reference architecture supplies a scalable and programmable paradigm for FC. The system's designer can apply their designed resource management plan in this reference architecture to enhance the LB (e.g., radio resource management, resource allocation, and task offloading) of the FC environment. Given the different use cases of FC, fog's LB needs to consider the exclusive features of the specific applications (such as automatic driving and data analytics) in the selected cases. The

computation task models we plan to introduce will be used for different applications. Here, we list some of the applications of FC security and transportation.

2.4.1.2. Real-time Data Analytics

Like IoT applications, the Internet of Vehicles (IoV) must reinforce its ability to tackle the high car production rate. With the advent of self-driving and semi-autonomous vehicles, sensing data (e.g., generated by sensors) in smart cars must be analysed locally in real-time and operations (e.g., directions and driving speed) must be taken. Even though smart cars might handle all the tasks required to analyse locally, the fog system can categorise the tasks that rely on the priority defined in the predefined stage or setup. This feature gives a chance to critical tasks for execution with minimal time, whereas the other requests are pushed to the cloud to be analysed or not taken immediately. For example, if the oil engine expires, the indication would be to act or store the historical data for that car.

Furthermore, many decisions require processing data that does not exist locally (in a vehicle). Hence, FC acts to accelerate this process and assist in making fast decisions at the edge. Many researchers study and propose many algorithms related to mobility computing integrated with FC. Hou, Li et al. (2016) investigate how Vehicular FC (VFC) might utilise vehicles as computational power and communication infrastructures. The research demonstrates an intriguing rapport between FC and computational mobility power, which can be used to empower that market. Chen, Huang et al. (2021) present CFC-IoV architecture, a regional fog-computing-based intelligent vehicular network architecture for dealing with big data that generates IoV objects. The proposed architecture includes infra-fog energy-aware and QoS-aware resource management, which aim to enhance the performance of the entire IoV system.

Smart Grids

The smart grid is a new era for an electricity distribution network. Usually, this network contains distributive smart meters installed near clients to count and measure real-time consumption for each unit. Supervisory control and data acquisition, known as SCADA, is a standard system utilised by utility organisations such as water and electricity to monitor and control distributive objects. These

objects are smart meters, sensors, control switches, or control valves. Traditional SCADA systems use telecommunication protocols to communicate over private or public networks, such as IEC 60870-5-101/103. This protocol has many obstacles when used via the internet. Smart meters leverage networking and cloud technology to communicate with a central unit in utility organisations. Therefore, IoT accelerates smart grids' ability to communicate with the central SCADA system over the internet.

FC is emerging in this field to enhance and accelerate the reaction of SCADA systems by analysing and processing tasks near the edge. Many researchers study smart grids in fog environments. Okay and Ozdemir (2016) presents a model for smart grids based on FC. The authors prove the efficiency of FC for smart grids compared to cloud systems. In contrast, Zahoor, Javaid et al. (2018) propose the HABACO technique to enhance resource management in intelligent grid systems by using the cloud or FC. The research shows the benefits of using FC in the SCADA system.

IoT Applications

IoT applications, such as smart grids and cities, increasingly rely on real-time data to operate efficiently. Moreover, with the incredible amount of data generated continually via sensors and other spread devices, the uni-data centre (such as the cloud) cannot be centralised in a data centre. An FC framework can compute the data aggregated from the IoT sensors near the source point. Therefore, this framework can handle both problems in this manner.

Although the cloud system has higher computational power than FC, fog/cloud computing has more power than the cloud. Traditional network components, such as routers, Access Points (AP), switches, and proxy servers, are available in the FC paradigm. Those components give FC an advantage over cloud computing by locally providing processing and analytic power. Thereby, FC can perform efficiently regarding service latency, network traffic, content distribution, and operational expenses. Therefore, FC better meets the IoT application requirements than using cloud computing.

2.4.1.3. Comparison of Cloud and Fog Computing Architectures

The main distinction between fog and cloud computing is the goal of minimising latency in the overall system. While the cloud centre resides apart from the field, some clients' apps require almost zero time to perform their tasks. Health applications, for example, are among the most prevalent applications that require low time latency, as are intelligent grids. The main differences between the two systems are listed in Table 2.5.

Table 2.5: The differences between fog and cloud computing.

feature	Cloud Computing	Fog Computing
Location	In promises	Closed to the clients
Internet Required	Crucial	Not essential
Technical Support	24/7	Client Responsibility
Globality	Support	Not support
Size of Recourses	Massive and easily expandable	Limited and not changeable
Owned by	Organisation	Clients

2.4.2. Load Balancing (LB)

In FC, LB is a strategy that makes all resources of the units' system equally used. It targets distributing the arriving tasks fairly to fully equip all active computing units. Every LB algorithm has a unique mechanism to accomplish this target. Generally, if managed well, LB is used to increase system throughput, resource utilisation, performance, and reliability. Although LB research inspired different solutions to fill most research gaps, an explosive growth of IoT data, users, and application requirements increased the burden on the fog layer, which has limited resources.

This section expands writing about LB, especially in offloading algorithms. In this field, various definitions of LB have been found. LB aims to maintain the available resources in computational nodes by redistributing the arrival tasks among computing nodes to be equally loaded, avoid overload conditions of one device (or specific nodes) with many tasks (Puthal, Obaidat et al. 2018), and avoid idle conditions for them.

Accordingly, LB in FC refers to efficiently distributing the arrival workload among a group of computing fog nodes, thus increasing the reliability of incoming tasks and the capacity of existing clients. LB is beneficial for cloud and fog providers to effectively distribute application tasks to different fog nodes or cloud servers. Due

to the fluctuation in the number of client requests, the LB technique is the most suitable technique for network computation, which works on a dynamic principle to avoid lagging in execution due to a long waiting list at one node. Among the previous definitions we notice, it seems they all have the same meaning. When offloading specific tasks, criteria such as excessive computation or resource constraints, latency requirement, permanent or long-term storage, data management and organisation, privacy and security, accessibility, affordability, feasibility, and LB are used.

Figure 2.2 depicts a model of LB in fog architecture, in which the cloud server interacts with different layers. LB process, the workloads are exchanged among layers to fill clients' needs regarding resources with the least possible delay. Further, in another part, the sensitive request gets sent to the cloud if the client fails to receive essential resources.

LB uses many metrics to evaluate, such as system performance, throughput, average waiting time, response time, make-span time, fault tolerance, network delay, execution time, resource usage, CPU usage, network load, and Quality of Service (QoS). The advantage of LB is that it utilises more computational and storage power while minimising the switching time between tasks. LB can be static or dynamic, periodic or non-periodic, and centralised or decentralised in a computing environment. Therefore, the main advantage of using an LB algorithm is enhancing the FC environment (Mon and Khine 2019).

There are two main categories for LB in FC: static and dynamic LB techniques (Bolarinwa 2015). Static LB relies on distributing the tasks using the initial information on task requirements. These requirements are determined at the beginning of the mission. Although this technique is easy to apply and configure, the static method has significant drawbacks. For example, they are changing the burden of one node during system start-up to the maximum load. Task allocation is fixed and cannot be modified during the execution of the process. While the network's initial design configures the distribution of the workloads to some nodes that seem not loaded, the nature of free movements in a fog environment will affect this strategy.

On the other hand, dynamic LB migrates tasks among nodes smartly when one of them gets too busy. Selecting the destination node always depends upon the current information about traffic loads. Hence, efficient real-time load estimation is vital for active dynamic LB. Moreover, most computing systems widely use virtual machines (VMs) (Pan, Thulasiraman et al. 2018). Therefore, the LB function might be more complicated in the fog environment.

Estimating the memory usage and CPU load for fog nodes and the network load is the first stage in the algorithm; secondly, analysing the available resources in fog devices and measuring the synchronisation level in fog zones. Third, notice the fault tolerance and avoid a single point of failure in the event of failure (Bisht and Subrahmanyam).

2.4.2.1. CLASSIFICATION OF FOG NODE IN LOAD BALANCING

Different methods have been proposed to classify the fog nodes. In the LB algorithm of FC, the algorithm classifies the nodes as heterogeneous or homogeneous.

Homogeneous Node: The FC environment usually compromises devices with different power capacities, such as a base unit with high power of resources, and, in other parts, mobile phones with a limited number of resources and a different type of constraining, so the fog environment is heterogeneous. Therefore, it is hardly adopted in the Fog environment. On the other hand, in heterogeneous nodes, this type of catalogue, the nodes agreed to cooperate with different types of nodes with varying power capacities. Therefore, the LB controller can choose the next execution node depending on specific conditions such as algorithm and task size.

Many IoT-generated datasets are processed in the fog layer. As mentioned, the fog layer comprises a group of fog devices, such as switches and routers, communicating to create a pool of computing and storage resources. These resources need to be managed by distributing the workload requested by clients. Those workloads sometimes have a high arrival rate at fog nodes and need fair distribution among processing nodes to benefit from those pools of resources. Thus, distributing loads requires an LB instant that uses a distinct distributing scenario. LB allocates loads to a suitable computing unit. A load balancer collects some information about fog devices to estimate and calculate the size of each device.

Based on this data, the workloads will be forwarded to the suitable fog node (Ferrer, Marquès et al. 2019).

A fog node is initially designated based on end devices' and traditional servers' features and characteristics. A group of fog nodes combines at least one or more physical machines with high computational capabilities (Ferrer, Marquès et al. 2019). An FC node would be a logical concept for a more robust understanding, with a device's different form as its physical infrastructure. All end devices, including processing capacity, will be counted within the fog node regarding virtual processing units. Therefore, all physical devices with computing power at a fog node are collected as a single logical unit able to process and analyse assigned services seamlessly. For example, traditional computers and mobile devices can also share their processing power to provide task orders. All coordination processes for the end devices, as well as communication issues within the fog node, will be managed by the fog node master (Mach and Becvar 2017).

2.4.2.2. Types of Controllers in Load Balancing

To create a balanced computing environment, a dispatcher must decide the required action to fulfil the balancing criteria. The primary role of the dispatcher is to investigate the surrounding statuses of computing nodes and select the ideal location to execute the arrival tasks. There are two strategies to implement LB in FC: centralised and distributed controllers.

The central controller uses a core device connected directly or indirectly to other fog devices. It could be a standalone or fog device with an LB feature. This feature is provided by modifying or configuring the existing computing nodes, as shown in Figure 2.4(a). For example, by configuring a router in a fog environment, it can act as a router and controller of LB at the same time. The pros of using a central controller technique for LB are that it is simpler to manage and implement. Moreover, the recovery time is minimal in the event of failure.

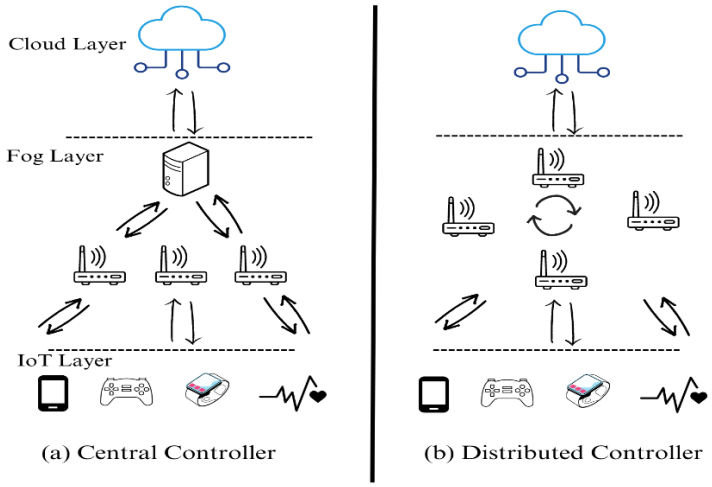


Figure 2.4: Type of controller in fog layer.

On the other hand, global state information in the network is a collection of metric information from other devices. Thus, the load balance device has global state information about the current cluster. This data is gathered and analysed in LB for redistribution (Chandak and Ray 2019). The cons of this technique are that there is a single failure point, and we can minimise the consequences by configuring another fog node to be a hot standby. On the other hand, global state information in the network is a collection of metric information from other devices. Thus, the load balance device has global state information about the current cluster.

Distributed Controller: A centrally located controller has many advantages in managing and controlling the load among devices in a network, but it might have a communication bottleneck since it utilises the network bandwidth. Thus, the failure of this load balancer will cause failure of the entire system, or more simply, a single point of failure. All fog devices must periodically send selective information to the central controller to sustain system updates. Therefore, cooperation is required between nodes to distribute the computation requests among them. Whereas the distributed controller, as shown in Figure 2.4(b), increases the scalability and reliability of the network, it relies on the cooperation of the local controller (Chandak and Ray 2019).

2.4.2.3. Scenarios of Load Balancing in Fog Computing

LB might occur at any computing level in the fog system, depending upon the location of the computing load. This location, which has a high demand for computing, determines what type of algorithms are suitable to be initialised and

executed, whereas every location has a different technique to apply. There are many offloading scenarios in the fog system.

Figure 2.5 illustrates some scenarios where offloading can be activated to achieve LB. There are three locations to offload tasks in the fog environment. For example, case A represents an IoT/cloud model where the IoT device communicates directly with the cloud platform without engaging the fog layer in communication, such as some intelligent meters accessing a cloud service directly. In case B, the communication line represents offload from IoT objects to fog level, where fog cooperates with the cloud to improve security or latency services or store and process data. While case C shows IoT devices accessing the smart gateway in a middleware, it evaluates the task. It decides whether to execute it locally or offload it to the cloud instead of offloading it to the fog, depending on the requirements of the service. In case D, the operation represents offloading among fog devices (peer-to-peer) after receiving it from an intelligent gateway or IoT objects.

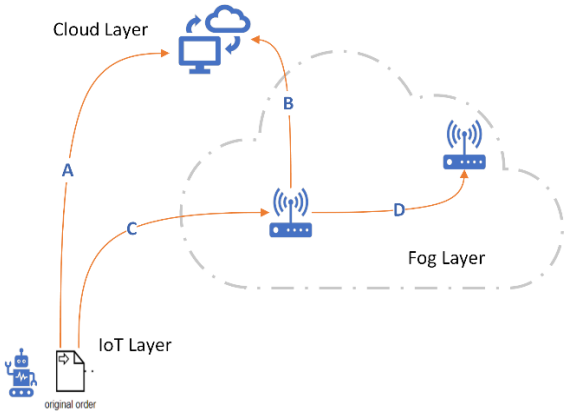


Figure 2.5: Possible offloading scenarios in a fog environment.

2.5. Challenges in Fog Computing

While FC is classified as an evolved extension of the cloud computing system to handle IoT-related problems and shortcomings at the network edge, in FC, processing nodes are distributed and heterogeneous. Furthermore, the services based on fog technology must work with various aspects of the restricted environment. Moreover, assurance of security is dominant in FC. Therefore, discovering the challenges of FC from service-oriented, structural, and security perspectives can be listed as follows (Mouradian, Naboulsi et al. 2017):

2.5.1. Service-Oriented

Resources enrich not all fog nodes. Therefore, comprehensive scale application enhancement in resource-restricted nodes is not natural compared to traditional data centres. Thus, distributed application development needs for potential programming platforms in Fog are required to be implemented. Moreover, a fog administrator must clarify the policies to distribute required tasks among sensors/IoT devices fog infrastructure.

2.5.2. Structural Issues

The infrastructure of FC consists of various components from both the core and edge of networks. These components have different computations but are not designed for general computing. Therefore, redesigning or modifying the computation unit for the element is a highly challenging part of the system setup. Additionally, based on execution operations and operational requirements, selecting the suitable device, places of deployment, and corresponding resource configuration are also crucial in FC. In FC, computational devices are spread across network boundaries and can be shared or virtualised. In this case, it is necessary to define suitable metrics, strategies for inter-nodal cooperation, and efficient resource provisioning.

2.5.3. Security Aspects

FC relies on conventional networking components, making it highly vulnerable to security attacks. Maintenance of privacy and authenticated access to computing and storage services in a widely distributed model, such as FC, is challenging to ensure. Therefore, maintaining QoS is difficult during the implementation of security, where the data centre integrity is adequate, which makes the security topic in FC challenging.

2.6. Discussion

Nowadays, a plethora of beneficial IoT apps rely on the quality of time-sensitive to run effectively. FC might be the appropriate technology to enhance the new generation of real-time apps. Because of the dramatically increasing number of IoT users, some computational nodes in a fog system become loaded while others remain idle. Thus, it has become essential to reinforce the cooperation and integration among computing nodes through offloading techniques to run most of the processing nodes equally and reduce the computational and storage power

wastage (in the idle nodes) (Janjic 2012, Nokia 2013, Bittencourt, Diaz-Montes et al. 2017, Mach and Becvar 2017, Baburao, Pavankumar et al. 2021). This chapter focuses on the research that has already been done on LB and the different ways to improve a fog model. Collectively, these studies outline the critical role and the need for offloading to enhance system performance, QoS, time latency, and power efficiency in FC (Vaquero and Rodero-Merino 2014, Shi, Pallis et al. 2019). Therefore, LB, through offloading processes, promises tremendous benefits in a fog environment.

At this stage, **the first three questions** of the research can be answered. After reviewing the recent study, the proposed solution must have the following characteristics to satisfy the research requirements:

- Utilising the behaviours of static and dynamic offloading in a novel offloading. (Answering RQ1).
- The proposed model must categorise the arrival client task into heavy and light tasks, as shown in (Guerrero, Lera et al. 2019). (Helping to answer RQ1).
- It is essential that light tasks be executed at the leaf or edge device and not be engaged in any offloading process (Sulimani, Sajjad et al. 2022). (Helping to answer RQ1).
- The novel hybrid algorithm, which has static behaviour, can work to minimise the exchanged system messages (as explained before); moreover, clustering limits the exchanged messages in a system by keeping them inside the cell. (Answering RQ2).
- Clustering also works effectively to limit the exchanged messages inside the cell. Distant offloading appears, in this case, when the fog node offloads the tasks to the cloud. (Answering RQ3).
- The proposed model must employ offload in cases C and D and avoid cases A and B (Figure 2.5). (Helping to answer RQ3).

However, the formidable rapport between enhancing the performance of computing nodes and offloading processes has been reported in the literature. In effect, the offloading technique reinforces the current algorithms and invents a new one (software). This chapter sets out to assess the drawbacks of offloading processes in the targeted system. Moreover, an initial objective of this study is to identify if any previous study introduced a sustainable concept to balance the load

in a fog system. The outcomes showed no LB solution based on a long-term concept with the mentioned characteristics in the literature reviews. These finding highlights adding an extra computing node to an affected area or moving the idle node to another location (hardware modification) with high demands to overcome the research issue. This modification in node position must be done through specific software, which must have historical load data for all computing points. However, creating a hybrid LB system is considered a measurable research gap. It can thus be suggested that even though the software solution has a noticeable improvement for LB, hardware and software resolution will enhance the computing system better. There is ample room for further progress in determining the feasibility of including hardware modifications in the position of fog nodes to improve LB.

2.7. CONCLUSION

Most suggested LB solutions that aim to reduce the workload of affected nodes by offloading some of that workload to their neighbours' nodes ignore the concept of sustainability in fog networks. This scenario is discussed and explored in this chapter. However, this perspective on resolution may lead to an increase in offloading processes, which would inevitably impact available network throughput. However, this chapter presents the most articles that offer solutions to enhance the LB in FC. The chapter uses a systematic review to build the survey. The research questions aim to understand better the proposed solutions to tackle the obstacles that affect the performance of time-sensitive applications in large-scale fog networks. Moreover, the study explores the sustainability concept utilised before. The outcomes of this chapter point to the fact that no study finding a solution to the research dilemma has a long-term solution.

3. CHAPTER, REINFORCEMENT OPTIMISATION FOR DECENTRALISED SERVICE PLACEMENT POLICY IN IOT-CENTRIC FOG ENVIRONMENT

Abstract: A decentralised service placement policy is critical in distributed systems like FC, where sharing workloads fairly among active computing nodes is essential. A decentralised approach is an inherent feature of the service placement process, may improve LB among computers, and can reduce the latency in many real-time Internet of Things (IoT) applications. **However,** this chapter proposes reinforcement optimisation for a decentralised service placement policy (RODSPP), which attempts to mitigate some of the drawbacks of existing placement policies. Matching task size with node specifications and allocating less popular but time-sensitive applications in the fog layer are the primary contributions of this study. Extensive experimental comparisons are made between the proposed algorithm and other well-known algorithms using an iFogSim simulator. A microservice-based application with varying sizes of computing requests is tested experimentally, and the results show that the proposed algorithm effectively serves computing instances closer to users, reducing service latency and network usage. Compared to the existing models, the proposed modified algorithm reduces service latency by 24.1%, network usage by 4%, and computing usage by 20%, thus highlighting positive outcomes when using the proposed algorithm for fog analytics in future real-time IoT applications.

3.1. INTRODUCTION

The popularity of smart cities, wearables, e-health, and intelligent vehicle environments has increased with the emerging applications of the Internet of Things (IoT) (Vögler, Schleicher et al. 2016, Sulimani, Alghamdi et al. 2021). Quality assurance is essential in intelligent systems because users and their service needs are growing. For the quality assurance of IoT applications, cloud servers were expected to integrate IoT technologies with human users (Catruc and Iosifescu 2020). The cloud-centric IoT network has been designed to alleviate the constraints of IoT devices due to the limitations of computational and memory resources by providing centralised control of IoT devices (Khosroabadi, Fotouhi-Ghazvini et al. 2021). However, cloud servers are often far from IoT edge devices, thus introducing latency and bandwidth challenges between users and computing nodes (Kamel, Yu et al. 2020). Latency in time-sensitive applications, such as smart grids or unmanned vehicle management, can result in significant problems (Kaiwartya, Abdullah et al. 2017, Tiwary, Sharma et al. 2018). A fog network can use local computing and storage resources near the IoT edge devices. It can thus provide necessary computations to the IoT edge devices, providing an opportunity to reduce the latency and bandwidth requirements due to being closer to the IoT edge devices than cloud servers (Klas 2015, Mao, You et al. 2017).

Cloud servers and fog networks provide alternative options for IoT edge devices that can also connect to access computing resources through cloud servers and fog networks. A fog data manager (FDM) can determine whether the data is executed in the fog network or cloud servers. The FDM can transmit part of the data to the cloud servers while the remaining data is retained on the fog network, as discussed by (Puliafito, Vallati et al. 2021). Once the data is sent and allocated to the fog network, the placement policy (PP) can ensure it is retained on the fog network. Fog service orchestrators (FSOs) can also fairly allocate computing resources among the fog nodes to improve the quality of service (QoS), as presented by (Baranwal and Vidyarthi 2021). However, neither of these methods addresses scalability issues in an IoT network well, and Salaht, Desprez et al. (2020) addressed scalability in their Fog Service Placement Problem (FSPP). An efficient FSPP is vital to ensuring the QoS

of IoT applications. A failed or suboptimal FSPP can delay fog analytics, causing significant issues for real-time IoT applications.

FSPP is essential, but its centralised management can have some disadvantages: (a) increased network overhead due to the periodic communications between the resource broker and fog node; (b) longer computing time due to the increased number of IoT edge devices to control; (c) reduced reliability to a single point of failure (SPOF); (d) management of heterogeneous fog devices; and (e) inherent latency generated by machine communications between brokers and fog devices. However, the decentralised scheduling algorithm (DSA) proposed by Salaht, Desprez et al. (2020) relies on the time vector for efficient operation. A poorly managed FSPP can cause real-time IoT applications to fail due to the 'induced' high latency (Brogi, Forti et al. 2020). A poorly managed FSPP can cause real-time IoT applications to fail due to the 'induced' high latency (Brogi, Forti et al. 2020).

3.1.1. Research Problem

"What is a new solution for optimal placement policy over fog physical resources to overcome the inherent issues of prevalent resource allocation mentioned in RQ1, RQ2, and RQ3? is the fourth research question. Creating a simple and efficient service placement model is the objective of this chapter. Several hurdles need to be overcome by the proposed algorithm, which requires developing a novel strategy in service placement to answer the related research questions.

This study reinforces the current placement policy FSPP using a proposed new approach to overcome some shortcomings. This study presents an innovative FSPP that operates distributed to reduce the induced and inherent latencies in IoT applications that prioritise delay-sensitive applications to be served with priority placement even with a low popularity rate. The father node, which receives the computing tasks from the end-user nodes, can forward or maintain the services if the father nodes are compatible. All tasks are managed locally by the father node and its subordinate nodes without global scheduling. Such an FSPP is more scalable and unaffected by the increasing number of IoT edge devices or services. This study also performs many experiments to confirm the above hypothesis. The primary contribution of this study is a more scalable FSPP that reduces computing

requirements while reducing inherent and induced delays in time-sensitive IoT applications. The new FSPP may limit global communication between fog nodes, therefore not guaranteeing global optimisation; however, as the size of the IoT network increases, global optimisation is often unnecessary (Puliafito, Vallati et al. 2021).

In this approach, the proposed method places the most popular or delay-sensitive compute tasks closer to the end-user nodes using the hop number as the reference. This method manages the workload for fog nodes to remain sustainable with locally optimised QoS, thereby satisfying the necessary computing requirements of IoT applications in the fog network. The primary objective of this study is to develop the decision criteria for service allocation, including when and where the services are to be allocated among the IoT fog nodes and associated time-sensitive requirements (either closer or further from the end-user nodes).

3.2. Related Works

Fog networks can use many optimisation techniques, including greedy algorithms, heuristics, genetic algorithms, and linear programming. In References (Yang 2015, Zeng 2016, Wang 2017, Khosroabadi 2021, Sami 2021), several aspects of fog resource management are defined, including scheduling, placement, provisioning, allocation, and mapping for clients, virtual machines, services, and resources. These algorithms have also been investigated for real-time applications, such as smart cities, industrial IoT, and mobile microclouds (Taneja 2017, Salimian 2021). Table 3.1 summarises recent work, including application types (e.g., eHealth IoT system, general IoT system, or embedded system) in the column Scope, types of brokers (Broker), proposed algorithms (Alg.), target functions to optimise (Objective functions), components that the optimisation algorithm can control to enhance the objective functions (Decision variables), validation tools used to test the algorithms, and types of computing environments (Env.).

Khosroabadi, Fotouhi-Ghazvini et al. (2021) proposed "a clustering of fog devices and requirement-sensitive services first" (SCATTER) algorithm to allocate the Fog-Edge border computing resources to delay-sensitive applications but ignores the other tasks in demand, which could be a concern in many real-life applications. Velasquez, Abreu et al. (2021) introduced popularity ranked

placement (PRP) based on graph partitions and optimisation using genetic algorithms and showed that PRP yielded improvements compared to the generic genetic algorithm or first fit (FF) algorithm but could still be improved. Morkevicius, Venčkauskas et al. (2021) presented two stages of multiobjective optimisation that included multiple metrics that we use in this study to evaluate the proposed method: security performance, compute usage performance, and storage utilisation. This technique was designed to maintain the QoS level with the minimum usage of available resources.

QoS and energy consumption metrics were used to evaluate the algorithms proposed by Puliafito, Vallati et al. (2021). Those authors attempted to enhance the service placement policy in a cloud/fog environment, ultimately improving QoS in delay-sensitive applications. Hassan, Azizi et al. (2020) proposed a MinRE placement policy to enhance QoS and energy consumption in a cloud/fog ecosystem. MinRE consists of two algorithms, MinRes and MinEng, each focusing on different workload types. During experimentation, MinRE showed promising results while increasing computing loads, even with a fixed number of fog nodes. Baranwal and Vidyarthi (2021) used distributive fog orchestrator nodes (FONs) to place services on a fog-integrated cloud. The FON, a mediator entity, was proposed by Al-Tarawneh to assign the arrival workload to the fog computational node (FCN). Tarawneh and Hyasat (2010) proposed a bi-objective placement algorithm to enhance the placement policy for interoperated services in a fog network. That study considered the security requirements and criticality of the application as optimisation variables and improved the satisfaction metrics compared to other policies, including edge-affinity and cloud-only.

Maiti, Sahoo et al. (2021) designed a service placement policy to use schedule gaps. Their proposed policy aimed to minimise the makespan to meet the task deadlines with less utilisation of communication resources. In (Sami, Mourad et al. 2021), an intelligent fog and service placement (IFSP) was proposed to proactively place services on demand using deep reinforcement learning (DRL) hosted in the cloud to predict the system's load expectations with the entire database. The IFSP can thus prepare nodes predictively before bearing workloads. Salimian Salimian, Ghobaei-Arani et al. (2021) proposed autonomous IoT service placement to reduce

the execution costs of a distributed fog system using the grey wolf optimisation (GWO) scheme, which outperformed comparable policies in finding suitable fog nodes to allocate computations. For heterogeneous cloud/fog environments, Arora and Singh (2021) presented a heterogeneous shortest module first (HSMF) placement policy. The HSMF was based on finding the shortest module to serve first, which yielded improved performance compared to the other approaches. Beraldi and Alnuweiri (2019) introduced a job allocation methodology for interdependent services in a fog network, where interdependent jobs rely on internode data communication and are described using a directed acyclic graph (DAG). This work was limited because it did not consider the deadline constraint and workflow execution. A linear approach (first comes, first serviced) has been widely used, but task completion in ascending order is more time-efficient (Jamil, Shojafar et al. 2020). However, sorting can be resource-intensive in terms of time and computing.

The decentralised service provisioning introduced by Guerrero, Lera et al. (2019) produced promising outcomes, such that all latencies in data transmission, decision processing, and return response were reduced, achieving fair LB. However, this system is still challenging due to its high running cost. Random service allocation was discussed by Souza, Ramírez et al. (2016) but resulted in a high service delay due to poor LB. Huang, Lin et al. (2014) proposed an energy-efficient approach to co-locating service placement, which was adequate for a small network but achieved poor service placement in a more extensive network. A similar approach by Wang, Zafer et al. (2017) reduced execution costs by placing predictive modelling tasks. The proposed solution performed well, but the cost of the system was high, making it unaffordable to a small company. Taneja and Davy (2017) proposed a resource-aware service placement solution, a good QoS optimisation approach that first considered the highest-demand application tasks. Thus, the mechanism resulted in a swamped network with idle and waiting states with low-capacity devices/nodes.

Table 3.1: Summary of approaches to service placement.

Authors	Scope	Broker	Alg.	Objective function	Decision variables	Val.	Env.
Khoshroabadi et al[13]	S	C-FON	SCATTER	QoS, response times, network usage, energy consumption, application loop delays	App. sensitivity	F, E	F
Velasquez et al[22]	G	C(Os)	PRP	Latency, jitter of app., application profile	Popularity of app	Y	CF
Morkevicius et al[23]	G	H(o3)	TSM	QoS, security, CPU performance, storage, energy consumption	Integer multiobjective particle swarm optimization	M	CF
Hassan et al[24]	S	C-FRM	MinRE	Response time energy consumption, resource usage	App. sensitivity	O	CF
Salimian et al[2]	S	H	GWO	Execution cost, average waiting time	Matching the task with node in the colony	M	CF
Baranwal et al[25]	G	D(Os)	FONs	Network relaxation ratio, processing time reduction	Enhance the selection of fog orchestrator nodes	M	CF
Zeng et al[18]	E	Ds	H	Min and max response time	Minimize computation and I/O time	O	F
Jamil et al[32]	H	C	SJF	Energy efficiency delay	Computing time, network usage	F	CF
Maiti et al[28]	G	H	SGAP	Average make span average schedule length	Latency	M	F
Al-Tarawaneh et al[27]	G	Ds	Bi-Obj	Application performance, energy efficiency	Sensitivity and security	F	F
Taneja et al[36]	G	Ds	MM	QoS	Use the border nods	F	CF
Deng et al[37]	G	Ds	H	Energy consumption delay	Save communication bandwidth	Mt	CF
Wang et al[38]	G	D	H	Resource utilization	Use the boarder nodes	O	ME
Guerrero et al[33]	G	D	POP	QoS, cost, execution time	Popularity of service	F	CF
Yang et al[17]	G	Ds	CSPP	Hop count, CPU, network	Matching the task with node	E	ME
Arora et al[31]	S	C	HSMF	Network usage	Popularity of services	F	CF
Sami et al[14]	S	C	IFSP	QoS, cost, execution time	Proactivity service	O	CF
The proposed method	G	D	RODSPP	Hop count, network usage	Popularity of service, task sensitivity	F	CF

Abbreviations: C, IoT-camera; E, embedded systems; G, general IoT system; H, eHealth IoT system; S, IoT smart home application (***Scope***). C, centralized; D, decentralized; Ds, distributed; H, hierarchical, O, orchestrators (***Broker***). Bi-Obj, bi-objective; CSPP, cost-aware service placement policy; EPOP, enhancement of placement optimization policy; FONs, fog orchestrator nodes; H, heuristic algorithm; HSMF, heterogeneous shortest module first; IFSP, intelligent fog and service placement; MM, module mapping; PoP, placement optimization policy; PRP, popularity rank placement; SCATTER, clustering of fog devices and requirement-sensitive service first; SGAP, schedule GAP's; TSM, two stage method (***Algorithm*** (***Alg.***)). E, experiment; F, iFogSim; M, Matlab; Mt, mathematically; O, their own simulation program; Y, YAFS (***Validation*** (***Val.***)). CF, cloud/fog; F, fog computing; ME, mobile edge-clouds (***Environment*** (***Env.***)).

3.3. Proposed Architecture and Algorithm

3.3.1. Architecture

In real applications, a fog network is an extension of a cloud network. The extension of the decentralised, remote distribution of the cloud network can be considered a fog network. The computing nodes within the fog network are equipped with limited storage capacity and computing power for host instances and can thus be regarded as cloudlets (Muniswamaiah, Agerwala et al. 2021). Therefore, a resource management policy for the IoT network can determine where and when to place

computing instances among the different layers of computing nodes (e.g., fog or cloud). This chapter proposes an innovative service allocation architecture that alleviates the drawbacks of FSPP, particularly for time-sensitive applications.

Given the three layers of the IoT network (cloud, fog, and edge), as shown in Figure 2.2, the top layer is the cloud layer that contains the primary cloud servers. The access layer is the edge layer, where end devices manage requests for IoT applications. The fog nodes connect to both the edge devices and cloud servers. Edge devices include computers, smart cars, sensors, and smart homes (Sabireen and Neelanarayanan 2021). The fog nodes connect to both the edge devices and cloud servers. This study assumes that all applications in an IoT network use microservices (Colistra, Pilloni et al. 2014). These applications were configured as a group of stateless and trivial services. These small services complete a complex task once executed in a sequence. For example, app_1 requires s_1, s_4, and s_6 to perform its complex task, while app_2 requires s_1, s_2, s_3, and s_5, where the system consists of N services, as shown in Figure 3.1. Both applications must complete a specific number of jumps to perform app_1 and app_2. Thus, each computing node can be scaled up and down in their instances to improve the QoS of the distributed fog nodes. The scaling process can use service codes from the cloud.

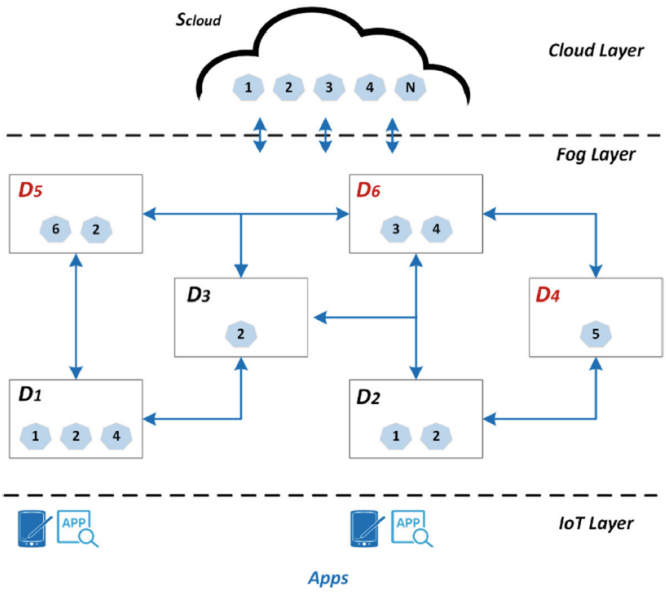


Figure 3.1: Interoperated services and applications.

This article proposes an optimisation of service placement by (1) allocating the most demanded services and all time-sensitive applications in a particular area in the fog nodes that are closer to the clients (edge); (2) migrating the services that are

not used regularly to reserve the available resources for other priority services; and (3) managing service allocations to prevent system overloads. For example, if an enormous computing task is allocated to a less powerful device, an overload will occur in the system. Due to resource limitations in fog nodes, the service placement policy must select services to be migrated. The proposed algorithm allocates priority services to the nearest nodes in the shortest path. It migrates the lowest nonsensitive requests to the upper levels, away from the users and closer to the cloud.

System latency is inherent in all IoT applications. Although the service placement policy can reduce latency in some fog applications, a delay-sensitive application demands further reduced latency to perform adequately in a time-critical application. For example, a device for a stroke patient must respond within near-zero seconds, where any delay in the reporting system can have marked consequences for the patient (Pareek, Tiwari et al. 2021). The metadata of service requests must be evaluated in priority assignment; thus, critical tasks can be allocated near edge nodes to support real-time requirements.

Many IoT applications perform interrelated services, where the cloud server becomes essential as a long-term solution. Migration of such services along the shortest path to the cloud is critical. The migration of some services can increase the total execution time (and thus latency) if some essential services do not migrate through the shortest path (Al-Tarawneh 2021). For example, an algorithm finds a service S_x with the highest demand for the device D_1. However, due to the limitation of $D_1's$ resources, s_2, which is selected with the lowest popularity in D_1, must migrate to neighbouring devices, as shown in Figure 3.2. Although, D_3 With a short distance from the source device, the number of hops increases by two for D_2, located on the shortest path to the cloud. To accomplish this task, s_4 was required.

Some services are in high demand (i.e., popular), while others are in low demand. The popularity rule works by migrating high-demand services to fog nodes closer to the client (edge), while other low-demand services are allocated to the upper nodes with the shortest path to the cloud servers.

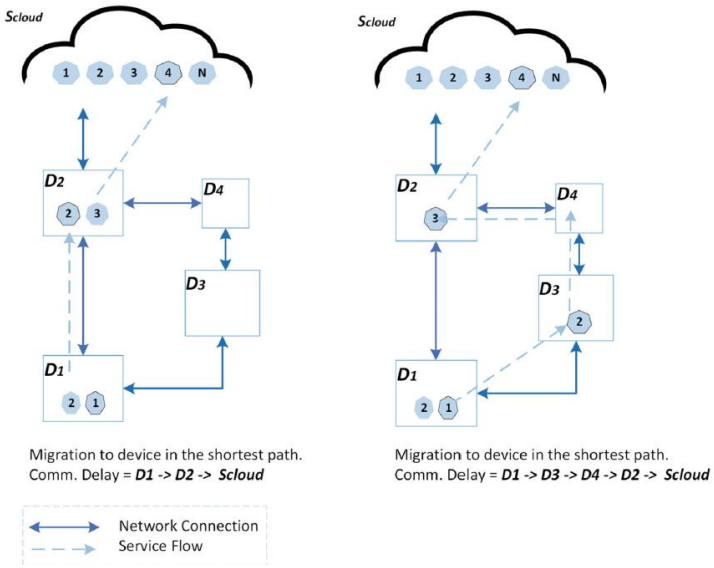


Figure 3.2: Example of service migration within different paths.

The interoperated service combines a clump of compensated microservices to a complex task. Thus, it is critical to migrate all interoperated services (if possible) if one must be migrated; this action tends to minimise the number of hops. Figure 3.3 shows the idea of partially migrating the interoperated services. We assume that the service execution flow for an application is $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4 \rightarrow s_5$. Due to this, it resides at the upper level, closer to the cloud and away from the edge. The application will add two additional hops if the placement management system (PMS) does not migrate to the following interoperated services: s_4 and s_5. Thus, it is essential to verify the migration of S_{n+1} and higher and keep s_{n-1} at the same node. Suppose we assume S_n is the migrated service, as shown in Equation 3.1. In that case, the PMS must maintain the migration process for the $(n+1)^{th}$ interoperated service along the shortest path to the cloud while keeping the $(n-1)^{th}$ in the initial node D_i:

$$migration_{s_n} = \bigwedge_{x=n}^{x \to \infty} s_x \quad \forall\, x \in interoperated\ services \tag{3.1}$$

This strategy must be activated if the initial node has limited resources. The computing node uses a decentralised service broker in the proposed system, which has proven beneficial against a centralised management system (Gowri and Baranidharan 2023).

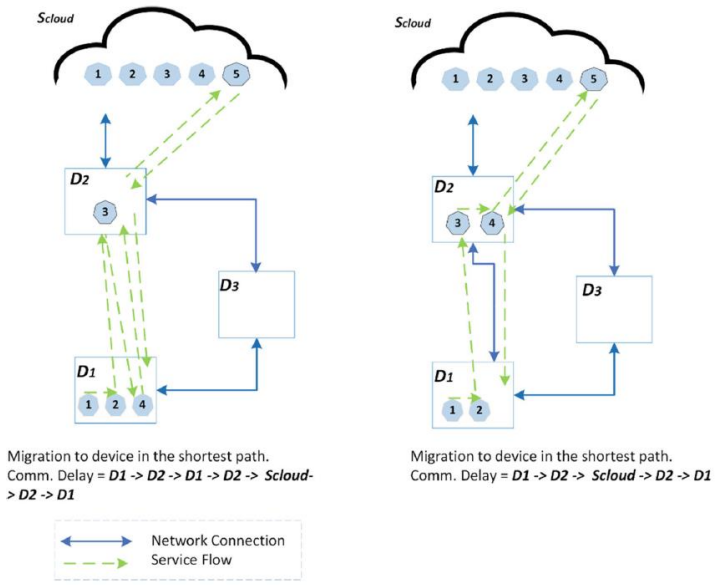


Figure 3.3: Example of interoperated service migration within the shortest path.

A decentralised broker is used to implement the proposed strategy. The modelling and characterisation parameters were obtained from (Besharati, Rezvani et al.) but with modifications. The modification is made using the service placement request manager (SPRM). A matching code is used in the SPRM to add more restrictions to accept the tasks or migrate them to the higher layer. Therefore, the computing nodes accept workloads by matching their sizes with node specifications. Conversely, Figure 3.4 shows the proposed broker, which keeps the remaining components of the broker having similar functions as Service Manager (SM), Service Usage Monitor (SUM), and Service Popularity Monitor (SPM) (Guerrero, Lera et al. 2019).

Fog clients must be connected to one leaf device or gateway in the fog network to start using available services. Each time the client requests a specific service, a service allocation request (SAR) is generated to obtain permission to reserve the computing resources. The SPRM decides to accept the allocation of this service or forward the SAR to the next upper fog device (see Figures 3.2 and 3.3). After analysing the local device information gathered from the SPM and SUM, the SPRM decides. Although the SUM aims to collect internal data about fog devices, the SPM is designed to measure the service request rate, which the SPRM uses in its calculations. Each node decides to host the services or migrates them to the next available node in the upper layer.

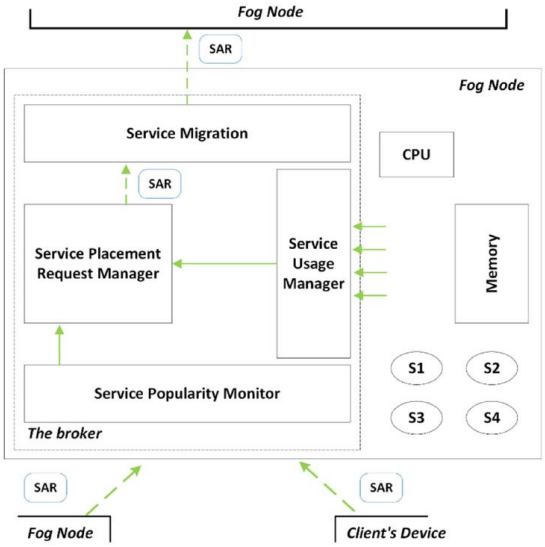


Figure 3.4: Decentralised service placement, the broker.

3.3.2. SYSTEM MODEL

In the proposed model, which follows the principles of FSPP, all applications requested by a group of clients, C_n, are initially allocated to cloud servers, S_{cloud}. Each application comprises a set of (interrelated) service units. Therefore, a sequence of service units must be executed to perform a specific application. Applications in the cloud are called by clients in the lower layer (closer to the edge) with a group of interconnected fog devices D in between. These fog nodes have constrained but available processing power that can complete the sequence of service units without relying on cloud servers. SP_S^{Cloud} is a term used to define the shortest path between service tasks and the cloud with the minimum number of hops. The father (D_1) of the device represents the first device that receives service requests.

The decentralised approach allocates services to local (neighbouring) fog devices. Various instances (S_x^y, where x is the service number and y is the code of the device that hosts the services) are allocated across the fog nodes. Thus, considering a given instance, we can formulate the relation as a many-to-one rapport $alloc: \{S_x^y\} \rightarrow \{D_x\}$. Conversely, many-to-many is the rapport if we consider $\{S_x\} \rightarrow \{D_x\}$.

To complete an application, every client C_x must be connected to the system through a leaf device. The leaf device can communicate with one or more devices

simultaneously. Thus, the rapport in this model was defined as a many-to-one rapport $conn.\{C_n\} \rightarrow \{D_x\}$. The service request rate $\gamma_{S_x}^{C_n}$ for each client must be considered in this study to categorise services. Each device D_i connected to the system must analyses its behaviour by monitoring the request rate $\gamma_{S_x}^{D_i}$ that it receives for each task. To calculate the request rate for D_x, for example, we must sum all the arrival tasks of clients who are in the range covering D_x as follows:

$$\gamma_{S_x}^{D_i} = \sum^{C_n} \gamma_{S_x}^{C_n} \qquad \forall\, C_n \in The\ coverage\ area\ of\ D_i \tag{3.2}$$

To determine the performance of each device in the system, we consider the computational capacity of a fog device. Therefore, the resource capacity is introduced as $R_{D_i}^{Cap} = \{r_{cpu}\}$, which is constant for each device and where $R_{D_i}^{Consp}$ is the power consumption of D_i depending on the allocation process in each device, which must be variable. Therefore, it is essential to calculate the total resource usage $R_{D_i}^{utz}$ for each computing node. The total resource usage can be calculated as follows:

$$R_{D_i}^{utz} = \sum^{S_x} R_{D_i}^{Cap} \times \gamma_{S_x}^{D_i} \qquad \forall\, S_x \in D_i \tag{3.3}$$

3.3.3. Optimisation Model

The proposed algorithm allocates the most popular (in-demand) computing loads closer to the client layer. It allows all nodes to accept computer loads if resources are available with the correct specifications. Thus, the service request rate is part of the acceptance metric in the SPRM module. The decision for each fog node is to locally analyse the service request rate before migrating less popular services to the cloud. The heavy tasks that overload the local nodes also migrate to the upper level. Guerrero (2019) notes that heavier tasks should be kept closer to the cloud provider. In both cases, the interoperated services migrated to the cloud once classified. The migration of the interoperated services with the undesirable service decreases the number of unwanted hops in the network, as previously discussed in Figure 3.3. Table 3.2 summarises the functions and variables used in the proposed model.

Table 3.2: SUMMARY OF THE FUNCTIONS AND VARIABLES USED IN THE PROPOSED MODEL.

	Notation	Description
Variable	S_x	Service S_x in the system
	S_x^y	An instance y of service x
	D_i	Specific fog device in the system
	S_{cloud}	The cloud server
	$R_{D_i}^{cap}$	Resource capacity of fog device
	$R_{D_i}^{consp}$ Rdi cons	Resource consumption of dog device
Function	AvaS (D_i)	Function that return the list of the variable instances in device D_i
	Install (S_x^y)	Function that download and install specifc instance on specific device
	Father (S_x)	Function that identify father device for service S_x
	Overh (D_i)	Function that return 30% of current resource for device D_i
	ShrtP(D_i)	Function that return the list of fog nodes in the shortest path to cloud
	TopRSev($\gamma_{S_x}^{D_i}$)	Function the service which record the lowest request rate in device D_i
	DAS (S_x)	Function that return true if the service is delay-sensitive application.
	nDSA (S_x)	Function that return the list of non-delay-sensitive application.
	Migrate (S_x^y)	Function that deactivate the instance S_x or several instances in D_i and send SAR request to next device to start migration process

Algorithm 1 shows the pseudocode for the proposed enhanced service placement policy. The placement algorithm is invoked when a particular service cannot serve appropriately with the local fog nodes and/or when the available capacity of a fog device is inadequate to satisfy the maximum service requirements. (Line 1). The latter condition ensures that all fog nodes in the proposed policy have sufficient capacity to run the most popular services. If these criteria are met, the gateway or leaf device D_i is the father of route D_{father} (Line 2). The father device is the closest computing node to the client's gadget. Thus, it is critical to maintain the highest-demand instances in a leaf device (Roig, Alcaraz et al. 2020).

Additionally, once the father device is selected, the first-child device D_{Child_1} must be chosen (Line 4) by identifying the shortest path to the cloud (Line 3). Thus, the first-child device D_{Child_1} receives SAR messages if D_i begins straggling. The service placement request manager (SPRM) in the child device decides to host the service or migrate it again by generating another SAR to its child while recalculating the shortest path. The child's decision depends on the service's popularity in D_{Child_1}. Typically, SPRM gathers essential data by SUM and SPM, where both are located within the same local broker. The reinforcement optimisation for a decentralised service placement policy (RODSPP) algorithm also ignores the remaining child in the route to the cloud, where each fog node recomputes the shortest path once the previous criteria are satisfied. This strategy reduces operational conditions' dependency on the remaining distant fog nodes.

Because this algorithm aims to improve service availability in the fog nodes, the system sometimes downloads the services from the cloud. To download the service S_x^y in the candidate fog device D_i (Line 6), the candidate node must have adequate resources to serve the requested tasks; the device overload must also be within the acceptable threshold, and the tasks must be recognised as part of delay-sensitive applications (Line 5). Eventually, the algorithm uniquely guarantees the maintenance of the overload of all computing nodes, while the POP algorithm focuses only on the guaranteed availability of services at the nodes (Guerrero, Lera et al. 2019).

The proposed algorithm then installs services with the highest rate (Line 8); otherwise, the candidate service is migrated to the upper level (Line 17). The migration process extends the computing capacity by alleviating current loads. If the service is recognised as a high priority, the interoperated services for the lowest service rate must be migrated together (Line 11). The number of spaces must be identified to create the required computation space to perform the top services (Line 9). The placement algorithm sequentially migrates the lowest and interoperated services to its child (Lines 12-14). Thus, the node begins by sending $SAR_{S_x}^{D_{Child}}$to its child. Every child identifies their child to communicate directly. The required services are downloaded once the migrated services are deallocated from the father device (Line 15).

We now consider the case study device D_i is currently allocated s_1, s_2, and s_4 in Figure 3.3. Due to emerging popular services and limited resources, SPRM may migrate s_2 as the lowest service usage. This action had several consequences. First, D_1 is assigned as the father device, and D_2 is assigned as the child, where the shortest route $D_1 \rightarrow D_2 \rightarrow S_{Cloud}$ has two hops, and the other route has one more step $D_1 \rightarrow D_3 \rightarrow D_2 \rightarrow S_{Cloud}$. Second, the most popular service requirements must be less than the currently available computing resources, and the node configurations must satisfy computing requirements; the latter condition avoids overloading the computing nodes. Then, we are obligated to migrate s_2 and its interoperated services s_4, and the first migration process occurred for s_2, followed by s_4. Finally, the required service triggers the download, while the migrated services commence finding another child host in the shortest path.

Algorithm 1. Enhancement of placement optimization policy algorithm

```
Result: Data Output to Actuator
Input: D_i, S_x^y
Start;
Generate tuples;                                   /*Data transfer from IoT layer to Fog layer*/

while S_x^y ∉ AvaS(D_i) || (R_{D_i}^{Consp} - R_{D_i}^{Cap}) < TopRSev(Y_{S_x}^{D_i}).size do
    D_i ← father(S_x)                                          /*Activation of the Algorithm*/
    SP_{D_i}^{Cloud} ← ShrtP(D_i)                              /*Identify the shortest path*/
    D_child1 ← SP_{D_i}^{Cloud}                                /*Identify the 1st child*/

    if (R_{S_x}^{Consp} × Y_{S_x}^{D_i} < (R_{D_i}^{Consp} - R_{D_i}^{Cap})) & ((R_{S_x}^{Consp} × Y_{S_x}^{D_i} < OverH(D_i)) || (DSA(S_x)))
    then
        D_i ← Install(S_x^y)                                   /*Install the required instance from cloud*/
    else
        if TopRSev(Y_{S_x}^{D_i}) then
            R_{D_i}^{ToFree} ← (R_{S_x}^{Consp} × Y_{S_x}^{D_i} < (R_{D_i}^{Consp} - R_{D_i}^{Cap}))
            S_n ← (low(Y_S^{D_i}) ∩ nDSA(S_x))     /*Select the non-sensitive service & low request
            rate*/
            interS[s_n^y, s_n^{y+1}, ..] ← interS(s_n^y);          /*Identify the interoperated services*/

            while R_{D_i}^{ToFree} ≤ (R_{S_x}^{Consp} × Y_{S_x}^{D_i}) do
                D_child1 ← migrate(∧_{k=n}^{k→∞} S_k^{D_i} ∀ k ∈ interS[ ])

            D_i ← Install(S_x^y)
        else
            D_child1 ← migrate(S_x^y)
```

Algorithm 1 shows the pseudocode of the proposed resource optimisation algorithm. The migration request is activated only when the requested service exceeds the currently available resources. We have provided a service for data classification using a lightweight module (Guerrero 2019). The service allocation request is sent to the gateway and then onto the fog layer. In this layer, the controller estimates the service requirements and sorts the devices accordingly. Thus, time and complexity are limited in sorting and job allocation. Devices that have the required capacity will be identified and allocated for computing in a predictive manner; thus, the tasks will be assigned to those nodes that have the ability and are available for operation. Also, complex tasks will be assigned to nodes that are not busy and have sufficient power to perform effectively. Simple tasks are thus assigned to busy nodes with less capacity. This approach minimises latency and power consumption, and reducing power consumption generally reduces overall operational costs.

3.4. Evaluation and Experimental Results.

3.4.1. Evaluation

We used the iFogSim simulator (Gupta, Vahid Dastjerdi et al. 2017) for the microservice-based simulation. The module placement class has been modified to evaluate the proposed model. This evaluation compared the POP (Guerrero, Lera et al.) with the edge based on iFogSim's built-in placement policy. Different scenarios were considered to evaluate the proposed algorithm by varying the parameters of the number of applications, fog devices, and users/clients. The configurations of the devices in the simulation environment are shown in Table 3.3, and the experiments followed the same configuration parameters in the POP study (Guerrero, Lera et al. 2019).

Table 3.3: The summary of configurations of experiments.

Device	Parameters	Units	Value
Cloud	CPU	MIPS	40 000
	RAM	GB	40
	Bandwidth	Byte/ms	10 000
	Link latency	ms	100
	Power	Watt	107
	Request rate	Req/ms	0.01
Gateway	CPU	MIPS	10 000
	RAM	GB	10
	Bandwith	Byte/ms	1000
	Link latency	ms	50
	Power	Watt	100
Fog device	CPU	MIPS	20 000
	RAM	GB	10
	Bandwidth	Byte/ms	2000
	Link latency	ms	100
	Power	Watt	100
	Request date	Req/ms	0.01

The experiments used a tree-based network topology to manipulate the number of devices in a system. Each fog device interacted with another fog device at the next level via the shortest path to the cloud. This forwarding process represents the migration activity for the lowest-priority task. For simplicity, the network device's behaviour was not changed and was fixed with the shortest path once identified. This proposal identifies the number of children at each level, as shown in Figure 3.5.

As discussed in Section 3.1, cloud servers are assumed to provide unlimited computational resources. The memory in fog devices was sufficiently large in these experiments; thus, we can ignore memory's immediate influence, which is the

current trend in fog devices: memory is rarely a limitation. Although computational capacity is the prime evaluation vector, network bandwidth was also considered to affect migration, which relies on data exchange between devices. As the number of migrations increased, the number of hops also increased. Thus, we considered the number of hops as the metric to evaluate network usage.

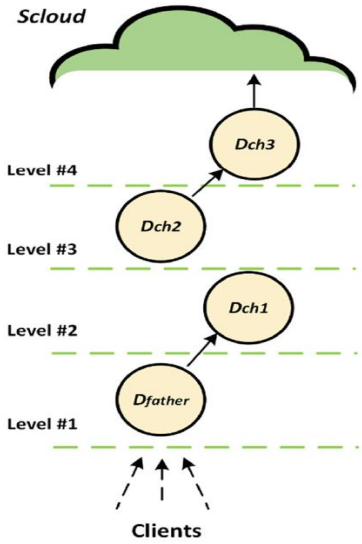


Figure 3.5: Proposed network.

Microservice-based applications are common in IoT domains. In microservice-based applications, the number of service placement requests describes service popularity. System latency is a critical factor for real-time services because delays may not be tolerated in some cases. The case considered in these experiments was an online store application, as shown in Figure 3.6. A similar benchmarking experiment is discussed in (Guerrero, Lera et al. 2019).

This microservice-based online store application is widely used in IoT modelling and is called a shock shop (Vögler, Schleicher et al. 2016). The configurations for this application in the proposed container followed the benchmark (Guerrero, Lera et al. 2019).

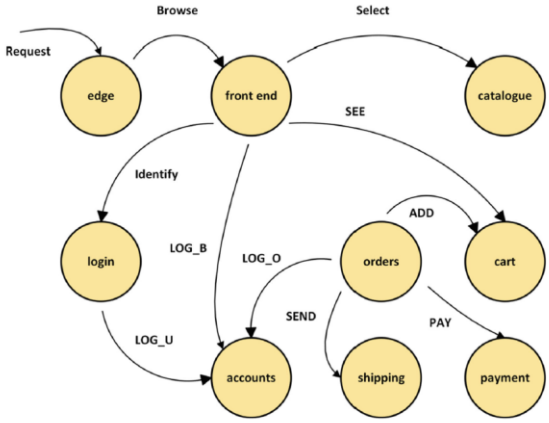


Figure 3.6: Application edges for online store-based case study.

3.4.2. EXPERIMENTAL RESULTS

The experimental setup for the proposed system followed existing recommendations. A different set of models was used in the experiment to compare and evaluate performance. The execution of simulation-based applications varies from machine to machine, depending on its scalability level. The details of the device model, operating system (OS), random access memory (RAM), hard disk drive (HDD), and processors that participated in the experiment are shown in Table 3.4.

Table 3.4: SYSTEM SPECIFICATION.

Device	Model	OS	RAM	HDD	Processor
1	HP	Win-10	6 GB	500 GB	AMD 2.9 GHz
2	Dell	Win-10	8 GB	1 TB	Intel 3.0 GHz
3	Lenovo	Win-7	4 GB	500 GB	Intel 2.56 GHz

CloudSim (Buyya, Pandey et al. 2009) is a requirement for iFogSim for cloud-based operations, and the data centre is managed by this tool. Java and JavaScript Object Notation (JSON) are used to program the proposed algorithm, while common math is the library used for complex mathematical computations in the simulations.

The experimental evaluation results are presented in terms of CPU usage, system latency, and network usage. The equations used to calculate these two metrics were programmed in the simulator. By analysing the iFogSim source code, we obtained the equation used to calculate them, as shown in Equation. 3.4 and Equation. 3.5, respectively:

$$Net_{Usg} = \frac{\sum^{Req_u(D_x,D_y)} \left(T^{ltsy}_{D_x \to D_y} \times Req_{Size}\right)}{Total\ simulation\ time} \tag{3.4}$$

where $T^{ltsy}_{D_x \to D_y}$ is the time consumed to migrate the request from one device to another and Req_{Size} is the size of the request that travels through the network. Network usage is the amount of data transferred from the original or the loaded device to the destination device, which enriches free resources at a specific time.

An increase in the number of devices leads to more complex network usage scenarios, that can result in network congestion, which indicates low system performance. A network usage comparison shows the low or high performance of a system. Compared to the edge policy, the proposed solution reduces network usage by executing most tasks closer to the clients and by a well-formed job allocation mechanism at the fog layer. This strategy minimises cloud usage, which eliminates the use of data transfer/communication networks by default.

System latency is the time required to perform a set of interoperated services to achieve the required application. iFogSim measures system latency by determining the average time needed to execute the complete path of the interoperated services. Latency is the most essential characteristic of the computing paradigm: the lower the system latency is, the more reliable the system. Equation 3.5 was used to calculate the system latency (Lty_{sys}) as configured in the simulation tool:

$$Ltsy_{sys} = \frac{\sum^{C_n} T_n^{Srt} - T_n^{Fnl}}{Req_u} \quad \forall\, Req_u \in \text{interoperated task} \tag{3.5}$$

where T_n^{Srt} and T_n^{Fnl} are the start and final times required by the nth service, respectively; and Req_u is the total number of requests in the interoperated service list. The following figures show a comparison of the three algorithms. The results of the proposed algorithm are labelled as RODSPP; those for Guerrero, Lera et al. (2019) are labelled as POP, and the edge is the label for the base policy of iFogSim. Figures 3.7-3.9 include two subfigures that show the effects of variation in the setting of execution: (a) variations in the number of users connected to one leaf device or gateway to examine varying levels of workloads; and (b) variations in the number of devices in the fog environment to study the influence of the route length on the network performance.

Figure 3.7 shows the hop count outcomes and plots the weighted average hope count proposed in the POP manuscript (Guerrero, Lera et al. 2019), representing how the most popular services are closer to customers. Figure 3.8 shows the latency results for a representative loop of the application. The experiment configured most parameters at their highest rates, such as accounts, orders, frontend, and edges. The simulator calculates the time the edge server takes to complete the requested tasks.

CPU usage is an important performance metric of a system because the processing quality of a system in a scalable manner is measured in terms of the CPU. If CPU usage is too high, delays occur during processing. The results of CPU usage, U_{CPU}, are determined by the following equation:

$$U_{CPU} = \frac{(T_b - T_i) \times R_t}{T_{mips} - F_{mips}} \times 100 \quad (3.6)$$

where T_b is the busy task rate, T_i is the idle task rate, R_t is the average tuple time, T_{mips} is the total number of MIPS (million instructions per second) in the host, and F_{mips} is the final MIPS in the control of the CPU.

Figure 3.9 shows the CPU usage of the devices by varying the number of users in each device. The experimental setup was configured as follows: two applications, up to five users per gateway, one child per device, and two fog devices per level. This setup is similar to the experimental benchmarking setup.

3.5. DISCUSSION

System latency is an important performance metric in FC. The weighted average hops effectively measure the proximity between the services and users, providing some sense of system complexity. Thus, we used both metrics to answer the fourth research question. Therefore, the series labelled Edge, POP, and RODSPP in Figure 3.7 were analysed. In Figure 3.7 (a), the graph shows a marked increase in the number of hops for all three approaches when the number of users increases, implying that the system would slow down once the number of users increases and the resources conflict. The weighted average hops indicated that RODSPP consumed 4% more hops than POP and 23.1% fewer hops than edges. The increase in hops in the RODSPP was due to migrating the interoperated services, which have low request rates and matching the task size with the node resources to avoid overload.

RODSPP prevented overload for the computing nodes in all layers by migrating excessive tasks from the overloaded ones.

POP and edge were not affected by the changes in the number of fog devices; RODSPP tended to migrate more services to the upper levels closer to the cloud. Due to adding two more service categories to activate the RODSPP algorithm, such as delay-sensitive applications, even with a low request rate, the migration process increased with an increasing number of fog levels in the system, as shown in Figure 3.7 (b). Therefore, RODSPP did not decrease the number of hops while maintaining the most usable and delay-sensitive services in the edge-fog network. This behaviour is acceptable if we try to solve many issues with limited resources in leaf nodes.

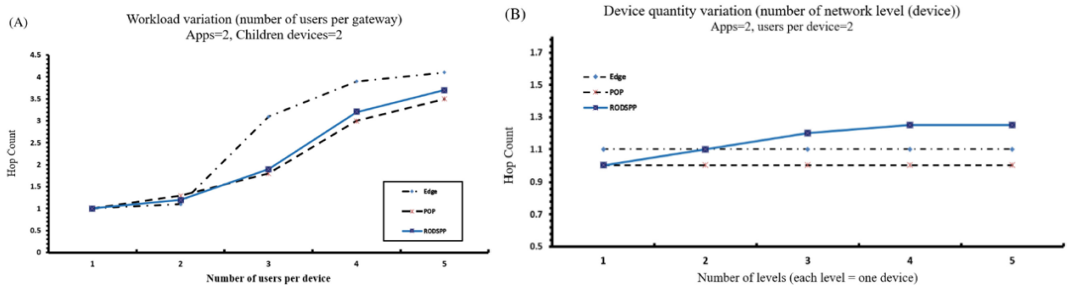


Figure 3.7: Hops count with different situations. Experiment with two applications, two users, and two levels of fog devices: (a) changing the number of devices associated with the edge device and (b) changing the number of proposed fog levels.

In Figure 3.8, after we split the tasks into high and low request rates, Figure 3.8 (A) shows that there has been a marked rise in the time latency incurred by the highest popularity applications. What is striking in the chart is the outperformance of RODSPP by 24.1%, which is due to matching the workloads with the capability of the computing nodes. Although RODSPP recorded a noticeable increase in hops, it achieved fair results in the overall performance scheme.

POP ignores low-requested services without considering their sensitivities. We added a delay-sensitive application with a low request rate to the proposed experiment. RODSPP allocates low-popularity applications on a leaf device if it is a delay-sensitive application. To answer the second research question, we consider Figure 3.8, which shows the performance of the POP and RODSPP for applications with low popularity but high time sensitivity. The chart shows the comparable behaviours of the RODSPP and POP policies in non-time-sensitivity applications. The chart shows that there has been a sharp decline in system latency for RODSPP

in delay-sensitive applications. Thus, despite its low popularity, RODSPP is a valid policy for delay-sensitive applications. Generally, even though the system places all services at the border with one user, the gateway still engages the cloud to accomplish complex tasks in real life. The third question in this study addresses the performance of RODSPP, among other models, in CPU, network usage, and system latency. The system latency parameter has already been discussed in the second question. The RODSPP showed outstanding outcomes for delay-sensitive applications with low request rates, as shown in Figure 3.8. Thus, RODSPP can be highly recommended for time-sensitive applications.

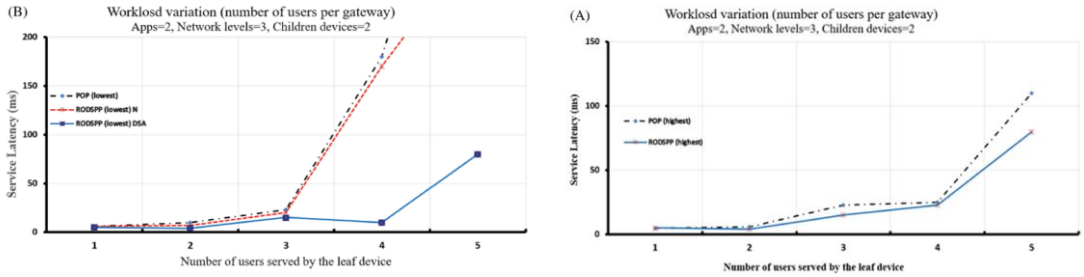


Figure 3.8: Service latency in different situations. Experiment with two applications, two users, and two levels of fog devices: (A).

Figure 3.9 shows the relation between the CPU usage in different layers and the network usage. Figure 3.9A shows the CPU usage of the father nodes in the proposed algorithm, which outperformed POP by matching the workloads with the available resources. This feature has improved the use of border CPUs. The proposed algorithm tends to decrease the load of children's CPUs once a new layer appears, which increases network usage. Although RODSPP consumes 4.5% more network bandwidth than POP, it decreases all fog resource usage by 20%.

This study's primary contribution was synthesising well-disciplined resource allocation approaches using them in local optimisation through a divide-and-conquer strategy in a carefully articulated realistic IoT fog networking environment. This study demonstrates successful IoT fog resource allocations in terms of controlled response time and constrained computing resource usage, thus providing significant insights and guidelines for the community to refer to and seek further enhancements in real-time IoT fog applications.

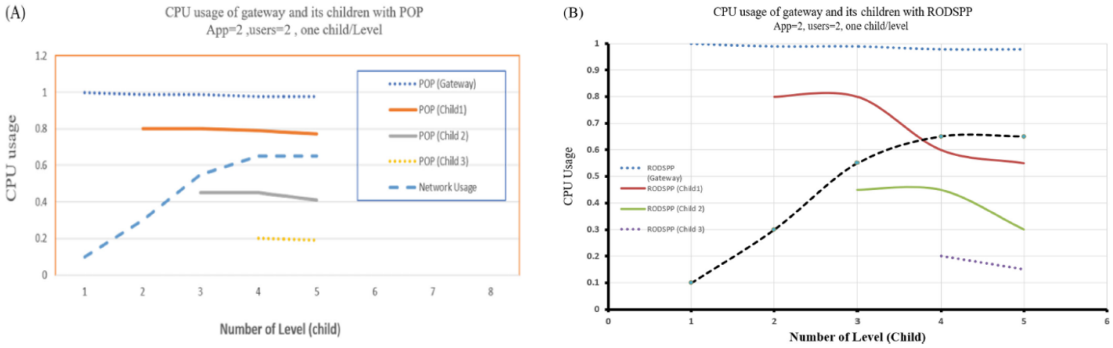


Figure 3.9: CPU usage of the devices regarding their topology distribution. Experiment with two applications and two users.

3.6. CONCLUSION

The span of IoT devices is growing daily, generating large amounts of data. However, due to its centralised and distant architecture, the cloud cannot manage and process an increasing number of IoT devices for real-time processing, which requires low latency and reduced resource consumption. Server mobility and decentralisation are requirements for IoT devices for real-time data processing. An FC paradigm is thus proposed in this study to meet the needs of future IoT networks. The proposed LB algorithm enables the FC mechanism to satisfy real-time performance requirements of the Internet of Everything (IoE), and the results demonstrate the scope of the proposed algorithm for utilising FC technology. This experiment paved the way for future evaluation.

4. CHAPTER, HYBOFF: A HYBRID APPROACH TO IMPROVE LOAD BALANCING IN FOG NETWORKS

Abstract: LB is crucial in distributed systems like FC, where efficiency is paramount. Current static offloading methods fall short in heterogeneous networks, necessitating dynamic offloading to reduce latency in time-sensitive tasks. However, existing offloading solutions often have hidden costs that impact sensitive applications, including decision time and distance offloading. This paper introduces the Hybrid Offloading (HybOff) algorithm, which enhances LB and resource utilization in fog networks, addressing issues in both static and dynamic approaches while leveraging clustering theory. Its goal is to optimize static offloading and enhance IoT application performance regardless of network size. Experimental results using the iFogSim simulation tool show that HybOff significantly reduces offloading messages, distance, and decision-offloading consequences. It improves LB by 97%, surpassing SOS (64%) and DOS (88%). Additionally, it increases system utilization by an average of 50% and enhances system performance 1.6 times and 1.4 times more than SOS and DOS, respectively. In summary, HybOff substantially contributes to LB and offloading research in FC.

4.1. INTRODUCTION

Central computing has emerged as a prevalent concept in various fields in the era of the internet, supported by 5G access networks. The central computing systems encompass technologies that empower enterprises to collect, process, analyze, and archive data from distributed clients worldwide (Datta and Bonnet 2017). This concept has become so integral to the internet that reverting to primitive (Peng, Zhao et al. 2023), decentralized systems are no longer feasible. In practice, cloud computing represents the tangible implementation of the central computing concept. It has gained widespread recognition as the ideal infrastructure for efficiently managing widely distributed Internet of Things (IoT) devices (Cheng, Wang et al. 2017). IoT, a telecommunication system facilitating data exchange among interconnected objects over a public network, streamlines operations with minimal human intervention (Lv, Wu et al. 2022). As a fundamental framework, IoT enables cloud computing to interact with the environment, facilitating the widespread adoption of IoT technology and the gradual growth of its data. However, it also presents implications for the efficiency of public networks (Khan, Laghari et al. 2022, Alsharif, Jahid et al. 2023).

Numerous critical applications rely on the same public network infrastructure, designed to support all cloud-connected applications (Cao, Li et al. 2021, Wang, Zhang et al. 2022). When slowdowns occur in the public network, time-sensitive applications such as e-health, smart grids, and unmanned vehicles, which have strict timing requirements for proper functioning, are severely affected (Jiang, Xiao et al. 2020). Cloud computing often struggles to consistently deliver the required level of service for these time-sensitive applications due to the unpredictable efficiency of public networks (Cao, Sun et al. 2021).

To address this challenge, Cisco introduced a new layer seamlessly integrated into cloud computing, forming FC. It integrates storage, computing, and networking at the network's edge, reducing data transfer to the cloud, lowering latency, and enhancing efficiency (Apat, Nayak et al. 2023). This technology is crucial for decentralized computing, especially in real-time IoT applications. However, the continuous growth in the number of IoT devices and their generated

data, along with the unpredictable nature of distributed IoT clients, places an increasing load on fog servers (Xu, Liu et al. 2023).

These factors drive researchers in FC to enhance the resource management system, particularly the LB system. It aims to allocate incoming tasks among servers with limited resources to prevent overloading or underutilizing fog resources. Effective LB management is vital to maintaining a stable computing environment and improving network availability and flexibility (Jiang, Dai et al. 2022, Goel and Chaturvedi 2023).

A steering algorithm is required to direct user requests to the most suitable fog server based on application requirements to achieve effective fog load balancing. Offloading is the primary mechanism for relieving overloaded servers, thus achieving LB in a distributed system (Goel and Chaturvedi 2023). A well-designed resource allocation policy is essential for creating an effective offloading strategy to balance load. In general, there are two fundamental approaches to offloading: static and dynamic (Cao, Zhang et al. 2021). Most recent offloading algorithms favour the dynamic approach due to its superior features to the static algorithm (Sulimani, Sajjad et al. 2022). However, dynamic offloading does have inherent drawbacks, including decision-making time, increased offloading messages, and distance-related issues (Xiao, Shu et al. 2022). These challenges result in significant network costs, often considered hidden expenses. Many articles view these costs as a trade-off for the reliability gained from dynamic approaches (Sulimani, Alghamdi et al. 2021).

The motivation for this research is rooted in the pressing need to address the formidable challenges posed by large-scale networks and time-sensitive issues, which, despite various studies on the subject (Meurisch, Seeliger et al. 2015, Li, Zhuang et al. 2018, Cao, Zhao et al. 2020), have yet to see a comprehensive solution that considers the hidden expenses associated with these challenges. This research introduces a novel approach, a hybrid algorithm, designed to simultaneously tackle these issues and ensure the selection of a suitable destination server for offloading. The imperative drives the impetus for this research to meet the escalating demands of time-sensitive applications in a world characterized by the continuous proliferation of IoT devices. Cisco's introduction of FC, which

seamlessly integrates storage, computing, and networking at the network's edge, is a notable development (Apat, Nayak et al., 2023). With its capability to reduce data transfer to the cloud, diminish latency, and improve efficiency, FC represents a significant step forward. However, the critical need remains for establishing an effective resource management system, particularly a LB system, to optimize the utilization of FC resources and establish a stable environment for time-sensitive applications. This research aims to develop a solution that simultaneously tackles the challenges of fog load balancing for large-scale networks, particularly in the context of time-sensitive applications. The research questions guiding this paper include:

1. How can fog load balancing be improved to support time-sensitive applications like e-health and unmanned vehicles efficiently?
2. What is the impact of offloading strategies on fog load balancing, and how can the hidden expenses associated with dynamic offloading be minimized?
3. Can a hybrid LB algorithm combining the strengths of static and dynamic offloading approaches provide a comprehensive solution to these challenges?

We introduce a hybrid LB algorithm that combines the strengths of both static and dynamic offloading approaches. The proposed algorithm offers five key contributions to fog load balancing:

- Reintroducing static offloading.
- Minimizing message exchanges.
- Reducing decision-making time for offloading.
- Encouraging servers to handle time-sensitive applications locally, eliminating the need for global allocation.
- Efficiently managing networks of all sizes using a cell-based approach, reducing latency, alleviating network congestion, and enhancing LB.
- Comprehensive experiments evaluate our algorithm from various perspectives, illustrating its superiority over other state-of-the-art fog load balancing algorithms in extensive studies.

Our work marks the implementation of the true essence of hybrid offloading, merging static and dynamic offloading behaviours. Additionally, the proposed algorithm incorporates various features, including a central-distributed control system, fog server clustering, and prioritization of critical applications while also

addressing hidden expenses such as distance-based offloading, decision messages, and network congestion. Compared to the static offloading scheme (SOS) and the dynamic offloading scheme (DOS). The experiments demonstrate that the proposed algorithm enhances LB by 52.1% and 38.2%, improves system performance by 60% and 38.8%, and increases the system utilization ratio by 62.4% and 42.7% compared to SOS and DOS, respectively.

The rest of this paper is organized as follows. The next section presents the literature review. Section 3 describes the proposed algorithm in detail. Section 4 shows the experiments and results, followed by the discussion and conclusion in sections 5 and 6, respectively.

4.2. Literature Review

In this section, the literature review explains the foundational concept of FC systems and the LB strategies devised to enhance offloading.

4.2.1. Fog Computing

FC, a pivotal concept in distributed computing, is engineered to efficiently support Internet of Things (IoT) applications, especially those demanding real-time responses (Mukherjee, Shu et al. 2018). As a complement to traditional cloud computing, it aspires to leverage edge resources strategically positioned closer to end-users (Das and Inuwa 2023). The core objective is reducing reliance on remote cloud data centres, reducing latency and decreasing network bandwidth requirements. Embracing FC presents various innovative advantages, including cost savings in cloud operations and fortified system stability (Burhan, Alam et al. 2023).

However, the continuous proliferation of IoT devices and the surge in data generation has strained FC's capacity to meet performance expectations (Alsharif, Jahid et al. 2023). This strain is particularly acute in specialized applications, especially time-sensitive ones. Varied growth rates in user density across different regions have resulted in an uneven distribution of workloads, causing some fog servers to become overloaded while others remain underutilized (Jebur 2023). This imbalance leads to resource wastage and misalignment within the fog layer (Jiang, Dai et al. 2022). To tackle these challenges, researchers have explored

dynamic offloading as a potential solution (Meurisch, Seeliger et al. 2015, Li, Zhuang et al. 2018, Lu, Gu et al. 2020, Tran-Dang and Kim 2023, Xu, Liu et al. 2023). Notwithstanding the merits of FC, due to inherent resource limitations within the fog layer, certain applications necessitate offloading to the cloud, emphasizing the enduring significance of web-based computing applications (Aazam, Zeadally et al. 2018).

To better comprehend the structure of computing networks in the proposed system, Figure 2.2 illustrates the three interconnected layers. Cellular or WiFi networks are wireless links connecting fog servers to client servers in the IoT edge layer (Jiang, Xiao et al. 2020). The internet is the primary medium connecting the fog layer and the cloud (Cao, Zhao et al. 2019). Within the fog layer, tasks are managed by surrounding fog servers, with results forwarded to the source server if necessary. The cloud layer is dedicated to specific purposes, such as heavy processing or data archiving. This research focuses on applications predominantly processed within the fog layer (Li, Zhuang et al. 2018).

All user-sent applications adhere to a common operational algorithm, as outlined by Mukherjee, Shu et al. (2018):

- Edge servers receive application requests from end-users.
- Received applications are decomposed into a set of sub-tasks for distribution.

Heavy fog servers either redirect the sub-tasks to idle fog servers for processing or add them to their processing queues. The processing results are subsequently sent back to the original server.

4.2.2. Related Works

The offloading technique is a pivotal solution for LB aimed at conserving computing and storage resources, particularly in decentralized systems (Tran-Dang and Kim 2023). A plethora of research efforts are dedicated to minimizing inefficiencies. However, prevalent task offloading schemes come with unavoidable hidden costs due to their specific requirements. These costs include offloading decisions, distance offloading, and network congestion (Wang, Han et al. 2023). Conversely, low-cost static offloading encounters numerous challenges, such as reliability concerns. This section delves into relevant publications and prior works

that validate the algorithm's novelty, which successfully addresses many of these obstacles.

In dynamic offloading, overloaded servers continuously gather data from other fog servers to distribute incoming tasks among the active servers (Tang, Xie et al. 2020). Once the system evaluates and processes this collected data, it makes an offloading decision, typically referred to as a 1-out-n process, where it selects the optimal target server (1) from among the available options (n) (Sethi and Pal 2023). However, this decision process leads to network congestion due to the periodic exchange of critical messages known as decision messages (Qu, Liu et al. 2022). In addition to network congestion, it also introduces high communication latency when identifying the target servers for offloading, termed decision latency (Zhang, Wen et al. 2022). While decision messages and decision latency may be minimal individually, they occur continually in affected areas, collectively impacting the effectiveness of dynamic offloading when following this approach.

On the other hand, the primary goal of most network operators is to 'serve more clients, earn more profit' (Jiang, Li et al. 2021). Expanding their coverage can increase the number of clients they serve, making coverage expansion a valuable metric for evaluating any network, as it correlates with increased network rank (Pavlovic 2008). While expanding the number of fog servers enhances system availability, it can have a negative impact on dynamic offloading. In such networks, dynamic offloading may offload tasks to remote servers, as many algorithms have no distance limits (Li, Zhuang et al. 2018, Jiang, Li et al. 2021). This can result in unfavourable outcomes, particularly for time-sensitive tasks, adding a burden on network bandwidth and total execution time due to messages travelling among remote servers (Meurisch, Seeliger et al. 2015). Therefore, distant offloading and offloading decisions hinder the effectiveness of dynamic offloading.

However, there are severe consequences if the fog system fails to deliver the expected services. Many critical applications that have recently emerged are time-sensitive, including unmanned vehicles, healthcare, and the smart grid (Gupta and Gupta 2022, Dhyani 2023, Kumar, Karunakaran et al. 2023). These applications rely on the fog layer for proper operation, where any delay can lead to catastrophic outcomes (Mutlag, Abd Ghani et al. 2023). Network congestion is another adverse

effect. The conventional offloading approach increases the number of messages in the network due to present system state requirements (decision messages) and distant offloading (in some algorithms). Consequently, the network infrastructure can deteriorate rapidly (Dhyani 2023).

Various LB algorithms and solutions have been proposed. In (Jiang, Chen et al. 2018), the authors introduce an energy-efficient offloading decision mechanism and an offloading dispatcher designed to balance energy consumption and response time for fog servers serving multiple applications in the IoT. This mechanism employs energy-aware cloud-fog offloading (ECFO), which aids in selecting the optimal target server with minimal utilization from the available servers. To address the issue of distant offloading and its associated consequences, ECFO assesses the cost of offloading decisions concerning bandwidth and energy consumption. This assessment is conducted through an energy-aware module by comparing it with the cost of local server execution. The proposed algorithm is evaluated against two state-of-the-art algorithms, and the results demonstrate that ECFO outperforms the others.

In (Ebrahim and Hafid 2023), the authors introduce a privacy-aware LB algorithm that employs reinforcement machine learning techniques to reduce the number of waiting tasks in the queues of fog nodes. The proposed algorithm, DDQN, does not rely on load or resource information from fog servers to determine the optimal server for offloading. Instead, it leverages Markov theory to estimate the availability of free servers. This approach significantly enhances system performance while maintaining privacy at an acceptable level. Interactive experiments demonstrate that DDQN outperforms a search-based optimization algorithm from the literature and traditional baseline approaches.

Albalawi, Alkayal et al. (2022) introduced a hybrid LB algorithm called PSOSVR, which combines particle swarm optimization (PSO) with support vector regression (SVR). PSOSVR reduces response time and energy consumption while improving resource utilization (RU) and throughput. The outcomes of this proposed algorithm notably enhance various metrics, with energy consumption improving by 56%. Using deep reinforcement learning, Lu, Gu et al. (2020) tackled the offloading problem in large-scale systems and multiple service clusters. Their

paper compares average execution time, latency, LB, and energy consumption, demonstrating that the IDRQN algorithm outperforms others. Tran-Dang and Kim (2023) proposed a dynamic collaborative task offloading (DCTO) algorithm to reduce execution time delays in fog systems. The algorithm has two main components: a task division technique and parallel execution. It seeks to identify the optimal target server for offloading among the servers in four layers. However, the algorithm does not prioritize sensitive applications over others.

In (Gowri and Baranidharan 2023), a dynamic energy resource allotment (DERA) technique that combines oppositional sparrow search (OSS) with the gravitational search algorithm (GSA) is introduced. DERA aims to improve energy efficiency and overall computing cost in FC environments, focusing on LB by reducing broadband costs, duration, and energy consumption. The proposed algorithm includes four layers: terminal servers, FC, cloud computing, and applications. The fog layer's controller module coordinates these layers. The DERA algorithm outperforms the DRAM algorithm by 6.96 percent in resource management through LB in most experiments. However, DERA does not prioritize sensitive applications and follows a centralized approach, which may limit flexibility and reliability.

Hussein and Mousa (2020) introduced two task offloading algorithms using nature-inspired meta-heuristic schedulers: ant colony optimization (ACO) and particle swarm optimization (PSO). They aim to minimize task response times while considering network latency, bandwidth, and fog server loads. Comparing these algorithms with the round-robin (RR) approach in extensive experiments, the ACO-based scheduler notably improves IoT task execution times. This ACO algorithm considers completion deadlines and optimizes fog server efficiency by finding the shortest path between the source and resources. However, it maintains some aspects of traditional offloading methods, relying on a central server for decision-making and processing time determination.

Lu, Wu et al. (2023) proposed a resource provisioning strategy to reduce the total mandatory cost in time-sensitive applications. The authors conducted a study in unlimited-processor and limited-processor fog nodes. Their paper introduces a heuristic algorithm that delivers exceptional performance in enhancing resource

provisioning, even under challenging conditions. Li, Zhuang et al. (2018) presented a Self-Similarity-based Load Balancing (SSLB) algorithm for large-scale FC systems. The authors introduced the concept of the 'cell,' which is sized to address distance offloading issues. While SSLB exhibits excellent performance compared to other algorithms, it does not offer advantages for Time-sensitive application (TSA), which has numerous restrictions. Additionally, the algorithm enforces uniform cell sizes, leading cells to be allocated to servers that may be located at a distance.

The previous section discussed various LB solutions summarized in Appendix I. These solutions primarily aim to mitigate the impacts of dynamic offloading rather than addressing the root cause of the problem. Despite their use of innovative technologies, they often entail hidden costs that can create an inconspicuous burden.

A summary of the current literature review reveals that dynamic offloading has gained widespread acceptance in FC. However, it is beset by inherent limitations, leading to significant consequences. Existing research has predominantly concentrated on improving dynamic offloading performance and catering to time-sensitive applications. Nonetheless, a noticeable gap exists in integrated solutions that can effectively address the inherent challenges of dynamic offloading, particularly those concerning offloading decisions and distant offloading.

This work aims to bridge these gaps and propel LB capabilities to new heights within the FC environment. To achieve this goal, it necessitates the development of a novel offloading strategy capable of surmounting these formidable challenges.

4.3. Hybrid Approach To Enhance Load Balancing

In this section, we dive deeper into the complexities of LB for Edge Computing and the innovative workload offloading solution we propose to solve these issues. Our proposed solution aims to directly address these challenges by providing an efficient offloading strategy that combines algorithms and real-time analytics to make informed task allocation decisions. By optimizing LB at the edge, we aim to

optimize resource usage, reduce latency, and provide a smooth and responsive experience for end users and devices.

As mentioned previously, many challenges and difficulties persist in fog load balancing, including distant and dynamic offloading issues, which drove us to create the hybrid offloading solution. The design of this proposed algorithm adheres to the following guiding principles, to address some of the shortcomings observed in prevalent algorithms:

- ***Flexibility*:** Given the decentralized behaviour of fog servers, where servers can randomly connect to or leave the fog environment, it is crucial to design a flexible mechanism that instantly reflects the status of connected and reconnected servers. Flexibility is enhanced by identifying a central server in each cluster that tracks clustered servers as they join or leave.
- ***Low Latency*:** Despite the minimal impact of offloading decisions individually, they occur continuously across affected servers. A novel offloading approach is followed to mitigate these effects, partly inspired by static offloading principles. The hybrid approach is crucial in minimizing the consequences of offloading decisions. Moreover, the proposed algorithm compels fog servers to serve locally, meeting the requirements of time-sensitive applications.
- ***Scalability*:** While increasing the number of fog servers enhances system availability, it can negatively affect dynamic offloading. The proposed algorithm mitigates distant offloading issues by grouping distributed servers into sets of cells. The clustering concept ensures that all adjacent servers interact with each other.

This work introduces the Hybrid Offloading (HybOff) algorithm, which aims to enhance LB efficiency and resource utilization in fog networks. The development of this hybrid offloading approach was motivated by the persistent challenges and difficulties outlined in the problem statement. Dynamic offloading mitigates these issues but has drawbacks: network congestion, high decision latency, and inefficiency with increased servers and distant offloading. These challenges are critical for time-sensitive applications like unmanned vehicles and healthcare. HybOff addresses these issues to provide adequate LB. Figure 4.1 illustrates the

estimated costs associated with prevalent offloading and the essential features it offers.

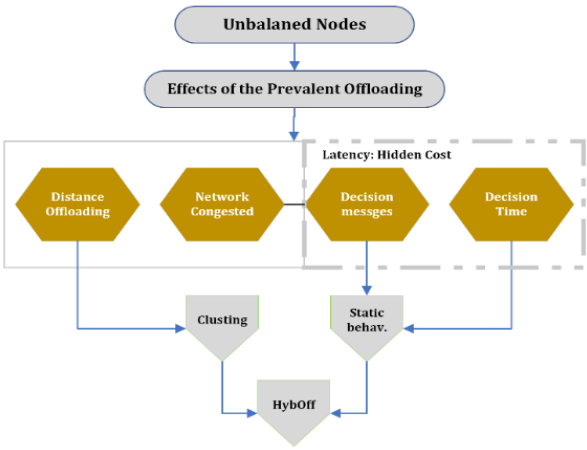


Figure 4.1: Prevalent offloading, costs, and solutions.

4.3.1. Problem Formulation and Terminology

At the beginning, Table 4.1 presents the essential notations used in this work, to facilitate the reading.

LB in fog networks demands innovative task allocation for end-user service requests, which are transformed into applications, underscoring the need for efficient task management (Zhang, Wang et al. 2022). In this context, this work assumes that the fog layer consists of a single level of fog servers with no vertical dimension, utilizing only the horizontal dimension for offloading. Additionally, we consider the fog layer to comprise W fog servers or nodes, denoted as Fn_1 *to* Fn_w, alongside n applications represented as cloud services.

$$App = \{App_1, App_{2,\ldots,} App_n\} \tag{4.1}$$

In this work, we categorize applications based on their task partitioning. Applications that are not partitioned into more than one task, and have time-sensitive requirements with deadlines, are classified as lightweight or TSA. On the other hand, applications with multiple tasks are categorized as Heavy applications (HA). Each HA, upon reception by the fog server, can be subdivided into a group of tasks, as shown below:

$$\mathrm{App_i} = \{\mathrm{AppTsk_{i1}^x}, \mathrm{AppTsk_{i2}^y}, \ldots., \mathrm{AppTsk_{iq}^z}\} \tag{4.2}$$

Each HA is divided into tasks, such as AppTsk_{i1}^{x} (the 1st task in the application *i* assigned to Fn_x), AppTsk_{i2}^{y} (the 2nd task in the application *i* assigned to Fn_y), and so on, with AppTsk_{iq}^{z} representing the q^{th} task assigned to Fn_z.

Table 4.1: ESSENTIAL NOTATIONS**.**

Symbol	Definition
TSA	**Time-sensitive application:** Refers to applications with strict time constraints, where processing and response times are critical.
HA	**Heavy application:** Denotes applications that require significant computational resources and are resource-intensive.
CPD	**Cooperating paired servers:** Represents servers that work in tandem or cooperation, often used for LB or redundancy.
SOT	**The Static Offloading Table:** A data structure or table that contains information about how tasks are offloaded from one server to another in a static manner.
W	**A complete set of system fog servers:** Refers to the entire collection of fog servers in the system, which collectively provide computing resources.
n	**Number of system applications:** Represents the count of applications within the system.
N	**Number of cells constituted after clustering:** Indicates the total number of cells formed after applying a clustering algorithm or process.
Q_k	**The queue of the k^{th} fog server:** Denotes the queue or waiting line for tasks that need to be processed by the k^{th} fog server.
Fn_i	**The i^{th} fog server:** Refers to the specific or i^{th} fog server in the system.
$\text{Fn}_i^{\text{CoD}_x}$	**The complementary server of x^{th} cell for Fn_i:** Denotes the server in cell x that complements or cooperates with the i^{th} fog server Fn_i.
$\text{Fn}_i^{\text{RU\%}}$	**The utilization percentage of the i^{th} fog server:** Represents the percentage of computational resources used by the i^{th} fog server Fn_i.
$\text{Fn}_i^{\text{M}_x}$	**i^{th} fog server which acts as a master of cell x:** Refers to the i^{th} fog server that serves as the primary or controlling server for cell x.
Cel_i^{Sz}	**The number of servers in i^{th} cell:** Indicates the count of servers present within the i^{th} cell.
$\text{Cel}_i^{\text{RU\%}}$	**The average utilization of the i^{th}cell:** Denotes the percentage of resources used, on average, within the i^{th} cell.
App_i	**The i^{th} application:** Refers to a specific application, often in the context of multiple applications running within the system.
AppTsk_{xy}^{z}	**The y^{th} task of application x computed in Fn_x:** Describes the task y within application x that is processed by the server Fn_x.
μ	**The theoretical difference between each consecutive server in SOT**: Represents the calculated or theoretical variance or difference between consecutive servers listed in the Static Offloading Table (SOT).
$\overline{\text{Sys}^{\text{RU\%}}}$	**The average system resource utilization:** Denotes the mean or average utilization of resources within the system.
$\overline{\text{Sys}^{\text{LB}}}$	**The average load balance of the system's cells:** This represents the average distribution of computational load among the cells in the system.

HAs are distributed to different servers for parallel processing once the partitioning process is completed. In contrast, TSAs are executed locally and receive the highest priority in the server's private queue, which is used to sort and re-sort received tasks.

$$\text{Q} = \text{Q}_1, \text{Q}_{2,\ldots,}\text{Q}_\text{W} \tag{4.3}$$

Tasks are generally queued on the system's servers when the server's computing power is insufficient to handle them immediately. For example, tasks from Appi are organized as follows:

$$\mathrm{App_i} = \{\mathrm{AppTsk}^5_{\mathrm{i1}}, \mathrm{AppTsk}^4_{\mathrm{i2}}, \mathrm{AppTsk}^7_{\mathrm{i3}}\} \quad (4.4)$$

It is important to note that application i is concurrently served by $Fn_4, Fn_5, and\ Fn_7$. In contrast, HybOff is designed to accept application subsets from a single server, reducing the load on network bandwidth. For instance, Fn_6 maintains tasks in its private queue, and it cooperates with Fn_9, as shown in Expression 4.5:

$$\mathrm{Q_6} = \{\mathrm{AppTsk}^6_{\mathrm{ax}}, \mathrm{AppTsk}^9_{\mathrm{bx}}, \mathrm{AppTsk}^9_{\mathrm{bz}}, \mathrm{AppTsk}^6_{\mathrm{ay}}\} \quad (4.5)$$

This proposed algorithm describes the workload as the number of tasks listed in the server queue for execution. Equation 4.6 shows the total time consumed for the workloads in the queue.

$$\mathrm{TET_{Q_i}} = \sum_{\mathrm{x} \in \mathrm{Q_i}} \mathrm{ET_{AppTsk^i_x}} \quad (4.6)$$

where, $\mathrm{ET_{AppTsk^i_x}}$ represents the execution time in milliseconds (ms) per task, with 'x' denoting its index. To improve the TET, a set of tasks ($\mathrm{AppTsk^i_x}$must be managed in each server queue, where the ET cannot be enhanced in this study (it is assumed to be fixed). Therefore, the LB issue can be addressed by efficiently redirecting the workload within each server's queue, as described in Equation (4.5).

Except for TSAs, this study employs a fixed-price algorithm for evaluating the available servers (Wu, Jin et al. 2022). Consequently, all servers have identical offloading costs. The system prefers to select a target server with sufficient resources, necessitating an evaluation process. Due to the homogeneity in server specifications, a suitable metric is utilized to identify the most appropriate servers for offloading. In the case of HybOff, the resource utilization percentage (RU%) for each server serves as an indicator to assess its available capacity, as computed in (Gupta, Vahid Dastjerdi et al. 2017). This metric depends on the computing power required to execute offloaded and local tasks. It's worth noting that HybOff does not factor in offloading costs in its calculations, as time-sensitive applications are executed locally (Cao, Sun et al. 2021).

4.3.2. HYBOFF DESIGN

In essence, control systems in multi-processing environments come in two forms: central and distributed. Central control, a traditional algorithm, suffers from reliability issues, as system failure can occur if the primary controller malfunctions (Sulimani, Alghamdi et al. 2021). Consequently, recent research favours distributed systems, where each computing unit functions independently. However, distributed systems lack certain central system advantages, like centralized server selection based on a comprehensive system analysis (Deepak and Pradeep 2012).

This work adopts a central-distributed control system as the optimal solution to combine central and distributed control aspects. It segments the extensive system into autonomous mini-controlled systems, forming the HybOff algorithm. This algorithm comprises interconnected computing cells, each housing a cluster of adjacent fog servers governed by an elected fog server known as the master fog server (Fn_i^M). Conversely, the other cell servers are referred to as followers. This design empowers Fn_i^M to monitor and supervise the performance of the followers, enhancing system flexibility. Even if a cell loses connection with others, each maintains an autonomous control system (Yang, Zhu et al. 2022). The interconnection of these cells forms the central-distributed control system, a framework that facilitates the implementation of HybOff, which requires multi-cells with distributed control.

In implementing the autonomous control system, each fog server has three modules: HybOffMonitor, HybOffComm, and HybOffSched. These modules handle monitoring, communication, and offloading, creating an independent control system for the fog servers, as depicted in Figure 4.2. As detailed in Table III, fog servers operate in two modes: basic and advanced. The advanced mode is activated in the master server, while the followers remain in the primary mode.

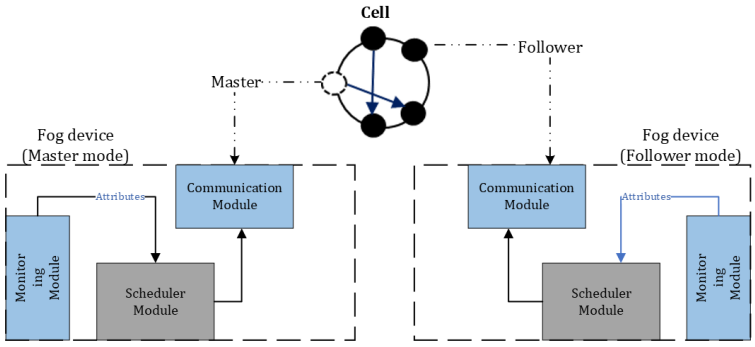


Figure 4.2: Architecture of HybOff algorithms. comprising three essential components: HybOffMonitor, HybOffComm, and HybOffSched, consistently maintained across all fog servers (Li, Zhuang et al. 2018).

In the basic mode, followers continuously use their monitor module to assess their workload and report it to the master server. The master server's scheduler module processes the data collected by the monitor module, determining the offloading policy needed for task allocation. The offloading process commences once the necessary information is disseminated within the cell via the communicator module. The communicator module is responsible for facilitating communication and message exchange among servers within the cell. The communicator module's thread is periodically generated to ensure all servers receive the necessary information. Additionally, it uses heartbeat information to address churn issues that may arise due to server crashes or new servers joining the network (Cao, Wang et al. 2021).

In summary, each master server collects workload data from the followers, processes it centrally, and then broadcasts the required offloading information to the cell servers to initiate static offloading. Table 4.2 shows the basic and advance features of each module.

Table 4.2: Features of HybOff Modules.

Module	Basic (Followers)	Advanced (Master)
Monitoring Module	Reporting the utilization percentage periodically.	Reading the utilization percentage of the followers periodically.
Comm. Module	Acknowledging and updating the target server for offloading process.	Maintaining static offloading table policy updating to create the list of targeted servers.
Sched. Module	Exchanging the server messages across the cell.	Working as a gateway to block internal messages within the cell and handle the outboard messages.

4.3.3. HYBRID FRAMEWORK

The HybOff algorithm's structure comprises a network of interconnected, distributed, and autonomously managed fog servers called cells. To initiate and operate the proposed algorithm, several steps must be performed:

1. ***Clustering:*** The concept of HybOff draws inspiration from the self-similarity load balancing (SSLB) structure, which forms segments (cells) of distributed fog servers with an equal number of fog servers (Li, Zhuang et al. 2018). Unlike SSLB, HybOff does not impose any restrictions on the similarity of cells; instead, the clustering algorithm selects cell members regardless of their size. As depicted in Figure 4.3, this approach ensures that cells are constructed using adjacent servers. However, significant benefits, such as reduced energy consumption and increased bandwidth, can be achieved if we confine adjacent servers in the same cluster to form cells, thereby minimizing communication with remote servers (Cao, Zhao et al. 2020, Guo, Zhou et al. 2022, Sofla, Kashani et al. 2022). Additionally, Li, Zhuang et al. (2018) suggest that servers within a close geographical area tend to exhibit similar behaviour, such as server joins or crashes (Jiang, Xiao et al. 2020). Therefore, initiating a federation of computing systems is crucial (Cao, Zhao et al. 2020). We employ the K-means algorithm described in (Likas, Vlassis et al. 2003) to build distributed cells, an algorithm known for its exceptional performance in large-scale environments.

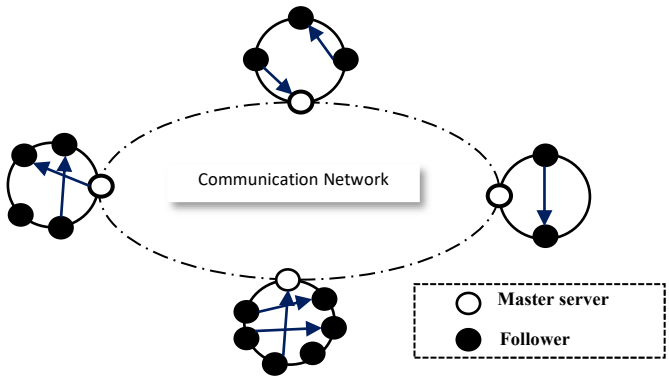
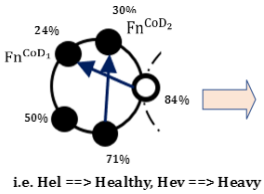


Figure 4.3: HybOff structure. In this centrally distributed architecture, contiguous servers are grouped as a cell. Each cell's.servers interact with one another while choosing a master for external communication and establishing a SOT.

The design defines computing servers as the set W, comprising m points in Euclidean space. The objective is to partition the W servers into N sets called cells ($Cel_1, Cel_2, Cel_3, ..., Cel_N$), each having a master. The variable cell size enables the K-means algorithm to discover the optimal server clustering. The size of any cell is defined as:

$$2 \leq \mathrm{Cel}_i^{Sz} \leq W \mid \forall i \in N, \tag{4.7}$$

where, Cel_i^{Sz} represents the number of fog servers in the i^{th} cell, which can be odd or even. For instance, in Figure 4.4, the system consists of 19 fog servers (W=19) as per the clustering algorithm, and they are organized into four cells (N=4). Each cell accommodates a different number of fog servers, as determined by Expression 4.7.



i	Fn_i	$Fn_i^{RU\%}$	$Status$	Fn_i^{CoD}	CPD	$CPD_i^{u\%}$
1	Fn_1	0.24	Hel	$Disb.$	1	ω
2	Fn_2	0.3	Hel	$Disb.$	2	$\omega + \mu_1$
3	Fn_3	0.501	Hel	$Disb.$	$Indv.$	$\omega + \mu_1 + \mu_2$
4	Fn_4	0.712	Hev	Fn_2	2	$\omega + \mu_1 + \mu_2 + \mu_3$
5	Fn_5	0.804	Hev	Fn_1	1	$\omega + \mu_1 + \mu_2 + \mu_3 + \mu_4$

Figure 4.4: Static Offloading Table in the Master Server.

2. ***Master Server:*** In each initialized cell, a controller server is randomly elected to oversee cell activities (Lu, Zheng et al. 2023). The master server (Fn_i^M) assumes various responsibilities, including:
   - **A.** Collecting resource utilization (RU) information from cell servers, including its data.
   - **B.** Updating the offloading table in the scheduler module.
   - **C.** Periodically broadcasting the required offloading information within the cell.
   - **D.** Monitoring cell servers to exclude any deactivated servers from the offloading process.
   - **E.** Serving as a gateway, connecting followers with external systems, and keeping exchanged cell messages confined within the cell, thereby preserving system bandwidth.

For example, if the master server in a specific cell fails, followers will lose external connections, such as those with the cloud and offloading functions. Nevertheless, followers can continue to perform essential computing functions until another master is selected.

3. ***SOT Policy and CPDs:*** Besides the HybOff architecture, the SOT policy plays a pivotal role in its design. A static table is inadequate for a system that requires flexibility. Therefore, SOT is a dynamic template within the master's scheduler module. It is unnecessary to offload all fog servers in the cell; instead, SOT

contains crucial cell data, including fog identification, $\mathrm{Fn}^{\mathrm{RU\%}}$, and the target offloading server. Fog identification is a unique number connecting to each server's Internet Protocol (IP) address, acting as a reference number. Each fog server corresponds to an individual row in the SOT.

Once the necessary data is available, SOT ranks active fog servers in ascending order based on resource utilization. This approach follows an ascending pattern, placing heavy servers at the end of the table and lighter servers at the top. After sorting the cell servers, SOT creates cooperating paired servers (CPDs). A CPD consists of two fog servers within the same cell with opposite resource utilization readings. The first server has the highest reading, while the second, a complementary server (CoD), has the lowest reading. This pairing is illustrated in Figure 4.4, where $\mathrm{Sys}_{\mathrm{avg}}^{\mathrm{RU\%}} = 54\%$, CPDs are formed by pairing opposite servers using Algorithm 1. Equation 4.8 specifies the servers participating in each pair, with $\mathrm{Cel}^{sz} - i + 1$ representing the index for the fog node paired with fog node i:

$$\mathrm{CPD_i} = \{\mathrm{Fn}_i, \mathrm{Fn}_j\}, \quad \text{where } j = \mathrm{Cel}^{\mathrm{Sz}} - i + 1 \tag{4.8}$$

Algorithm 1: Building and Maintaining SOT in the Master Server.

```
Output: Fn_i^{CoD_x}
Input: ∀ Fn ∈ Cel_a, Cel^{RU%}, Cel_a^{Sz}
Start;
mid = approx(Cel_a^{Sz} / 2);
Do
    If (OddSize(Cel_a^{Sz})
        disabOff(Fn_mid)                          //Disable the offloading function
    for (i = 1: mid; i++)
        CPD_i = (Fn_i, Fn_{Cel_a^{Sz}−i+1})       //Creating a CPDs
        BroadCast(Fn_{Cel_a^{Sz}−i+1}^{CoD_i})    // Update CPDs
    End
End
```

4. ***Broadcasting:*** After the creation of CPDs, the master server broadcasts complementary server information throughout the cell. The "broadcast()" function informs cell servers about their complementary servers. In contrast, the middle server (in the SOT when N is odd) disables the offloading function to operate independently without participating in the offloading process, achieved through "disabOff()". Light followers must also disable the offloading

function to prevent the system from entering a thrashing state. In a thrashing state, all servers spend time forwarding tasks among themselves without executing their primary functions (Kaur and Sachdeva 2020). The HybOff algorithm avoids this state by employing the "disable" function, which restricts specific and unnecessary servers from forwarding tasks. However, the "disabOff()" function only prevents servers from offloading within the cell, allowing them to continue offloading outside or to the cloud when necessary.

5. ***Static Offloading:*** Heavy servers initiate offloading as soon as they receive information about their complementary servers (Algorithm 2). They forward heavy tasks using the Last In, Last Out (LIFO) procedure, with priority given to all TSAs in their queue. Servers continue to utilize their complementary servers until they receive updated information from SOT.

ALGORITHM 2: STATIC OFFLOADING (ALL SERVERS)

```
Output: Provide destination server
Input: Cel^RU%, Fn^RU%
Start;
Do (Fn_a^RU% ≥ Cel^RU%)
    Fn_b^CoD ← Offload(Fn_a, HA(AppTsk_xy^a)        //Static Offloading
End
```

HybOff requires verification that heavy servers surpass the average load of the cell. In this algorithm, offloading occurs independently within each cell once a server is categorized as heavy. To establish the appropriate categorization criteria for servers, the average utilization ratio of each cell must first be calculated. Equation 4.9 provides the formula for categorizing each cell:

$$\mathrm{Cel}_i^{\mathrm{RU\%}} = \frac{\sum_{x=1}^{\mathrm{Cel_i^{Sz}}} (\mathrm{Fn}_x^{\mathrm{RU\%}} \times \mathrm{Fn}_x^{\varphi})}{(\mathrm{Cel_i^{Sz}} - \sum \mathrm{Fn}^{\varphi})} \tag{4.9}$$

where, $\mathrm{Cel}_i^{\mathrm{RU\%}}$ represents the average utilization ratio for cell i, and Fnφ is 1 if the fog server is active and 0 otherwise. The cell servers will not initiate offloading until the categorization criteria are met. In this algorithm, if $\mathrm{Fn_i^{RU\%}} \geq \mathrm{Cel^{RU\%}}$, Fn_i is considered a heavy server; otherwise, it is categorized as a healthy server. This condition deactivates the algorithm when all servers are not overloaded. For example, if all cell servers have a low load, no offloading process will commence, and each server will manage its workload locally. Thus, we can define this cell as a balanced cell, a feature that significantly benefits network bandwidth.

Let's consider an illustrative example to comprehend the relationships among cell servers. In previous Figure 4.4, if the clustering algorithm forms a cell with five fog servers, let's assume that the first server, after ranking in the SOT, has a utilization percentage of $\mathrm{Fn}_1^{\mathrm{RU\%}} = \omega$. It's important to note that there are variations in the utilization percentages among the sequentially ranked servers, denoted as $\mu_1, \mu_2, \mu_3, and\ \mu_4$ in our calculations. In this example, we have two CPDs, CPD1 and CPD2, each with a unique utilization reading. However, to calculate the RU for the ith pair, we need to apply the following relationship:

$$\mathrm{CPD}_i^{\mathrm{RU\%}}(i, \mathrm{Cel}_i^{\mathrm{Sz}}) = 2\omega + \sum_{a=1}^{i-1} \mu_a + \sum_{b=1}^{\mathrm{Cel}^{\mathrm{Sz}}-i} \mu_b, \quad (4.10)$$

where, $\mathrm{CPD}_i^{\mathrm{RU\%}}$ represents the utilization percentage of CPDi in the cell. Using Equation 10, CPD1 contains Fn1 and Fn5, while CPD2 contains Fn2 and Fn_4. When the load reaches the average cell load, Fn_4 and Fn_5 will offload their workloads to Fn_1 and Fn_2, respectively. Fn_3 operates independently as it has an adequate load. In cases where the number of cell servers is even, all servers are included in computing pairs. The utilization percentage in each pair is as follows:

$$\mathrm{CPD}_1^{\mathrm{RU\%}} = 2\omega + \mu_1 + \mu_2 + \mu_3 + \mu_4. \quad (4.11)$$

$$\mathrm{CPD}_2^{\mathrm{RU\%}} = 2\omega + \mu_1 + \mu_1 + \mu_2 + \mu_3 \quad (4.12)$$

Unfortunately, there is no mathematical relation that can predict μi. For simplicity, we assume that the utilization value between each sequential server is constant $(\mu_1 = \mu_2 = \cdots = \mu_{(sz-1)} = \mu)$. If so, we can conclude that: $\mathrm{CPD}_1^{\mathrm{RU\%}} = \mathrm{CPD}_2^{\mathrm{RU\%}} = 2\omega + 4\mu$, which represents the utilization percentage for any CPD in the previous example. In other words, HybOff equalizes the loaded pairs cell-wise. This work predicts the RU% for the cooperative pair servers using the following formula:

$$\mathrm{CPD}^{\mathrm{RU\%}}\left(\mathrm{Cel}_{\mathrm{i}}^{\mathrm{Sz}}\right) = 2\omega + \left(\mathrm{Cel}_i^{\mathrm{Sz}} - 1\right)\mu \quad (4.13)$$

Mathematically, all CPDs in the cell have the same load. However, the load of CPDs depends on the number of fog servers in the cell. For example, Cel_a and Cel_bcontain 6 and 13 servers, respectively, after the clustering algorithm builds the cells. According to Equation (13), $\mathrm{CPD}_a^{\mathrm{RU\%}} = 2\omega + 5\mu$, and $\mathrm{CPD}_b^{\mathrm{RU\%}} =$

$2\omega + 12\mu$. This means the shared computational load for each CPD increases with the cell size.

4.3.4. THE PROPOSED ALGORITHM

The identified drawbacks will be effectively addressed by integrating the cell concept within our hybrid offloading framework. In this design, Fog servers are structured into cells, where each server pairs up for resource sharing. Our proposed algorithm aims to maintain consistent average load levels across Fog servers within each cell, and you can visualize the algorithm's flowchart in Table 4.5.

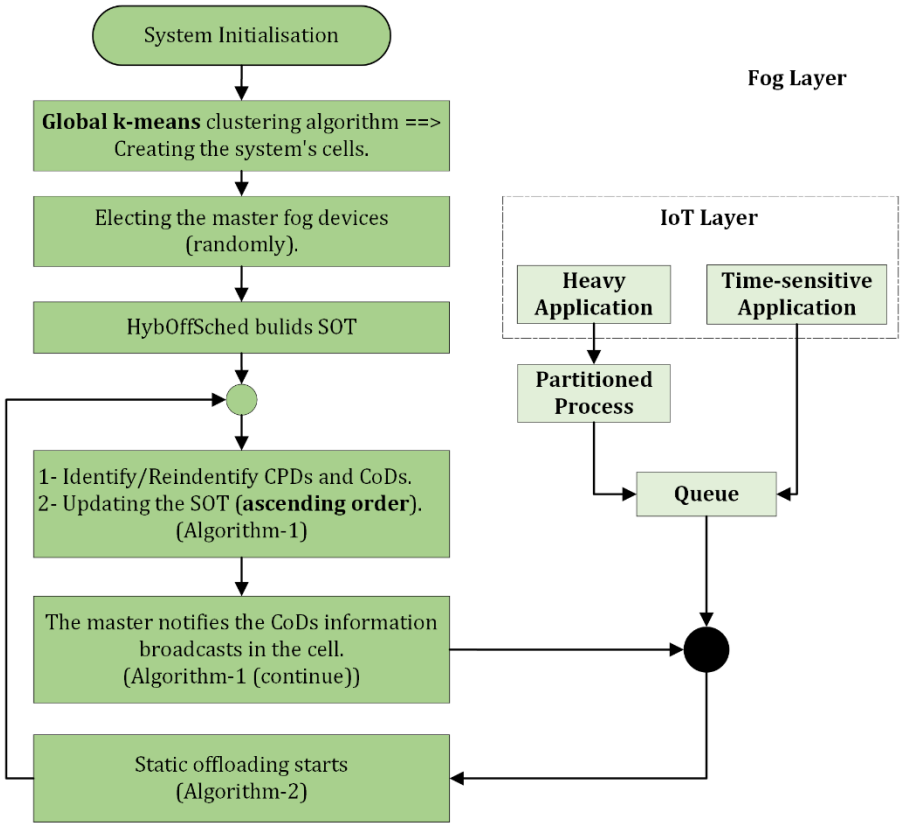


Figure 4.5: HybOff Process Flowchart.

As depicted in Figure 4.1, this hybrid LB algorithm capitalizes on the strengths of static and dynamic offloading strategies. Our proposed algorithm brings five crucial enhancements to fog load balancing:

- Reintroduction of Static Offloading: We're reintegrating the efficiency of static offloading into our approach.
- Minimal Message Exchanges: Our algorithm minimizes message exchange between servers, streamlining the LB process for greater efficiency.

- Reduced Decision-Making Time: We've significantly reduced the time required to make offloading decisions.
- Local Management of Urgent Applications: Our approach encourages servers to handle urgent applications locally, eliminating the necessity for global allocation.
- Efficient Network Management: We employ a cell-based approach for network management, reducing latency, alleviating network congestion, and enhancing overall LB.

The subsequent section illustrates these improvements through a series of comprehensive experiments.

4.4. Experiments And Results

4.4.1. Preliminary Explanations

This section assesses the proposed algorithm and demonstrates how the hybrid offloading structure outperforms other classical LB schemes. Generally, the essential requirement of effective LB is to keep all the computing units equally loaded by avoiding overloaded or underloaded cases (Sulimani, Alghamdi et al. 2021). The RU% of servers are used to evaluate the effectiveness of LB.

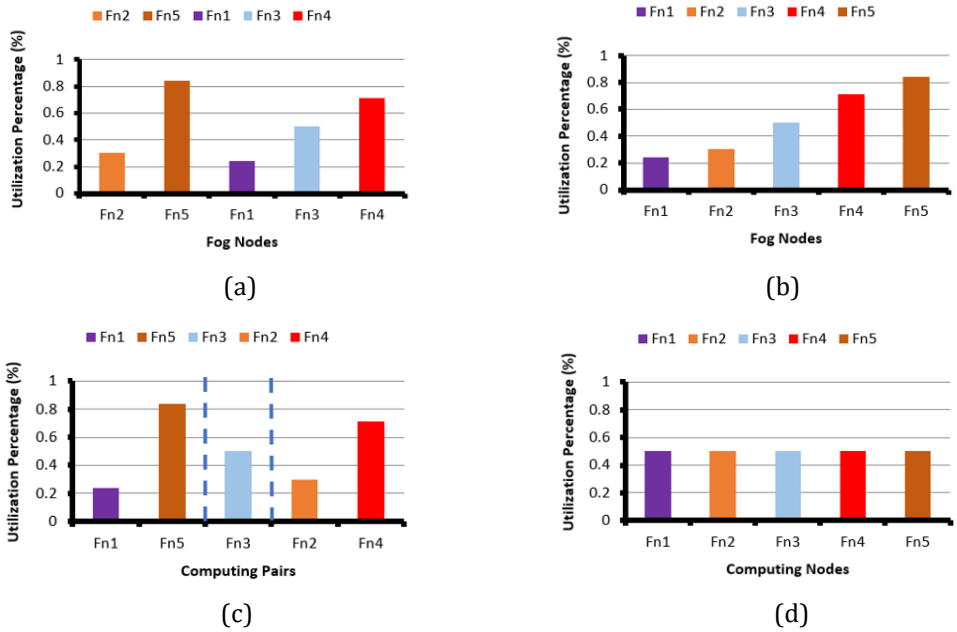


Figure 4.6: Balance of resource utilization-mathematical, (a) before offloading, (b) sorted servers, (c) paired servers, and (d) after offloading.

To demonstrate the efficiency of HybOff, we consider the example depicted in Figure 4.6 for a mathematical analysis. As shown in Figure 4.6 (a), the scheduler module collects the $Fn^{RU\%}$ for cell servers with a fixed difference (μ). The servers in the cell are ranked in ascending order, and the master creates the CPDs as depicted in Figure 4.6 (b) and (c). These figures illustrate how opposite servers share their load while the middle server operates independently. After a period of offloading, all cell servers have the same load, as shown in Figure 4.6 (d). Using Equation (13), we find $CPD^{RU\%}$ = 2ω + 4μ for each pair, where the $Fn^{RU\%}$ for the cell servers is ω + 2μ. Fortunately, the middle server also has the same load of ω + 2μ. HybOff achieves LB by dividing the cell servers into multiple pairs and ensuring an equal load distribution through sharing. HybOff successfully balances the load and creates balanced cells by ensuring that opposite servers share the load. On the other side, three metrics are employed to evaluate the proposed algorithm: 1) resource utilization ratio of the fog system, 2) loading balancing resource usage among fog servers, and 3) system performance. Resource utilization measures the usage of all the distributed fog servers' computing resources. LB determines the distributed tasks among computing servers in the fog layer. The system performance checks the efficiency of the entire algorithm.

4.4.2. Environment Description

Simulation Setup: The experiment follows the algorithm described in Figure 2.2, as outlined in the work by Lu, Zheng et al. (2023). It consists of W distributed fog servers and N created cells. Cloud services, denoted as 'n', are available on all fog servers, and offloading is initiated only in cases of computing power shortage. Tasks of varying sizes are processed on the fog servers. The initial experiment settings are summarized in Table 4.3:

Table 4.3: Initial Parameters of Experiment.

Parameter	W	n	ω	μ	ET/task
Value	Up to 300	15 apps	18%	7%	3 ms

It's important to note that this experiment focuses exclusively on the fog layer and does not consider the cloud. The simulation tool is iFogSim, which creates the necessary environment (Gupta, Vahid Dastjerdi et al. 2017). The experiment assesses various parameters across different server scales and data sizes (Cao,

Zhao et al. 2019) and examines resource utilization over time, considering random combinations of data sizes and scales, as detailed in Table 4.3 and 4.

- **Fog Server Specifications:** The specifications of the fog servers used in the experiment are provided in Table 4.4:

Table 4.4: SPECIFICATIONS OF FOG SERVERS.

Fn_i	Capacity	RAM	CPU
Fn_1	100 MB	7 MB	120 MHz
Fn_2	150 MB	15 MB	80 MHz
⋮	⋮	⋮	⋮
Fn_i	200 MB	10 MB	100 MHz

- **Evaluation Metrics:** To evaluate the algorithm's performance, we measured resource utilization (RU) in the described case studies using three different schemes: the static SOS, the DOS, and our proposed HybOff scheme. DOS is adapted from (Li, Zhuang et al. 2018), while SOS is configured using classical static offloading. The resource utilization ratio of the system in the experiment is calculated using Equation (4.14), where certain function components from the HybOff algorithm were modified and reused to implement SOS and DOS:

$$\overline{\mathrm{Sys}^{RU\%}} = \sum_{b=1}^{\mathrm{N}} \frac{\sum_{a=1}^{\mathrm{Cel}_b^{\mathrm{Sz}}} \mathrm{Fn}_a^{\mathrm{RU\%}}}{\mathrm{Cel}_b^{\mathrm{Sz}}}, \text{if } \mathrm{Fn}_a^{\mathrm{RU\%}} \cong \mp 5\% \ \mathrm{Sys}_{\mathrm{Avg}}^{\mathrm{RU\%}} \tag{4.14}$$

- **Task Specifications:** The specifications of the tasks used in the experiment are detailed in Table 4.5:

Table 4.5: TASK SPECIFICATIONS.

Process	Process size	Partitions	Sensitivity	Priority
P1	5 MB	1/1	TSA	High
P21	6 MB	1/3	HA	Low
P22	6 MB	2/3	HA	Low
P23	9 MB	3/3	HA	Low
⋮	⋮	⋮	⋮	⋮

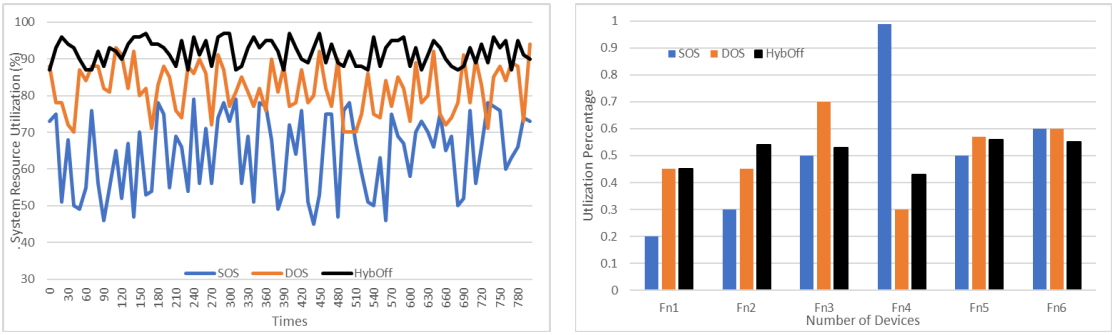
4.4.2.1. RESOURCE UTILIZATION

In this work, the resource utilization ratio of the system ($\mathrm{Sys}^{\mathrm{RU\%}}$) refers to the ratio between the number of resources utilized and the total amount of system resources. The used resource is any processor of a fog server that consumed more than or equalled the average cell utilization of its processing power. To do this, the RU must be calculated at every detection time using $\mathrm{Sys}^{\mathrm{avgRU}}$ (Equation 4.15).

$$\mathrm{Sys}^{\mathrm{avgRU}} = \sum_{i=1}^{t} \mathrm{Sys}_i^{\mathrm{RU\%}} / t\,, \tag{4.15}$$

where t represents the number of detection times during the experimental period, Figure 4.7 (a) illustrates that the system utilization ratio fluctuation is lower for SOS and DOS. This is primarily because HybOff enforces cooperation among opposite servers, enabling the system to tap into previously unexplored resources and communicate directly with the most affected servers to offload their load. In contrast, SOS and DOS experience inefficiencies in redistributing workload, resulting in a leakage of fully utilized servers. Equation 16 presents the formula used to calculate the system's utilization during the experimental period.

$$\mathrm{Sys}^{\mathrm{RU\%}} = \frac{\sum_{i=1}^{N} \mathrm{Cel}_i^{\mathrm{RU\%}}}{N} \tag{4.16}$$



(a) (b)

Figure 4.7: Resource Utilization over time and changing number of servers. (a) Resource Utilization of the System with Changing of Time, and (b) Resource Utilization Percentage.

To evaluate HybOff's efficiency in utilizing available resources at the server level, we need to determine the average RU under different fog scales and varying data growth rates. The same amount of data is generated for different cell scales to achieve this. The average RU provides insights into the algorithm's effectiveness in leveraging the available resources. Figure 4.7 (b) depicts the RU of fog servers in a single cell containing six fog servers. The experiment involved continuously increasing data generation without exceeding the half-capacity of fog servers in the cell. Figure 4.7 (b) illustrates the captured RU of cell servers at specific times. While all prevalent algorithms achieve approximately 76% utilization of edge resources, HybOff maintains an average of 50% in the cell. This indicates that HybOff evenly distributes the workload among the fog servers, unlike SOS and DOS, which fall short. HybOff's advantage stems from the clustering technique, which divides the fog servers into mini fogs. This approach allows HybOff to treat each cell as a mini-fog system, making it easier to manage and control. Additionally, hybrid offloading enhances RU further.

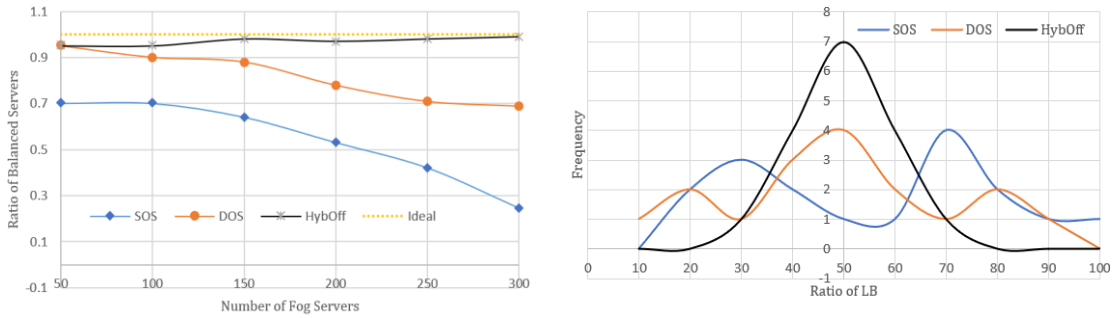
4.4.2.2. LOAD BALANCING

To assess the effectiveness of the proposed algorithm, this section evaluates the level of balanced RU among servers in the fog layer. It compares it to SOS and DOS, with the target level defined in (Sulimani, Alghamdi et al. 2021) where all fog servers were equally loaded. LB is the percentage of healthy fog servers in the cell, with a ±5% threshold value of ($Sys^{RU\%}$). In this experiment, however, we considered any server close to the average system utilization as a healthy server. To do this, we need to count the healthy server's cell-wise during the experiment, which satisfies the criteria previously mentioned. Equation 14 is used to calculate the RU for the HybOff algorithm, while Equation 4.17 is used to calculate the average RU for the SOS and DOS.

$$\overline{Sys^{RU\%}} = \frac{1}{N}\sum_{a=1}^{N} Fn_a^{RU\%}, \text{if } Fn_a^{RU\%} \cong \mp 5\%\ Sys_{Avg}^{RU\%} \tag{4.17}$$

Figure 4.8 (a) depicts the percentage of fog servers classified as balanced across various system scales, with experiments ranging from 1 to 300 servers, all using a fixed data size. The graph underscores HybOff's ability to consistently maintain a high percentage of healthy servers, closely aligning with the ideal curve. At 150 fog servers, SOS, DOS, and HybOff achieved 64%, 88%, and 97% for balanced servers, respectively. Impressively, HybOff continued to perform exceptionally well even with 230 fog servers. However, the dynamic scheme's performance deteriorated when the number of fog servers reached 300, revealing communication overhead as a bottleneck.

The performance of the static approach exhibits a decreasing slope, consistent with its strategy. Nevertheless, the results indicate that HybOff excels in large-scale networks, primarily because it is fragmented, and the central-distributed approach makes it easier to control and maintain. In contrast to the theoretical estimation of HybOff, which suggests effective load equalization among all computing servers, the experimental results do not align with this mathematical estimation. This discrepancy arises from the variable and uncontrolled nature of μ. The uncontrolled differences among consecutive servers diminish the performance of HybOff.



(a) (b)

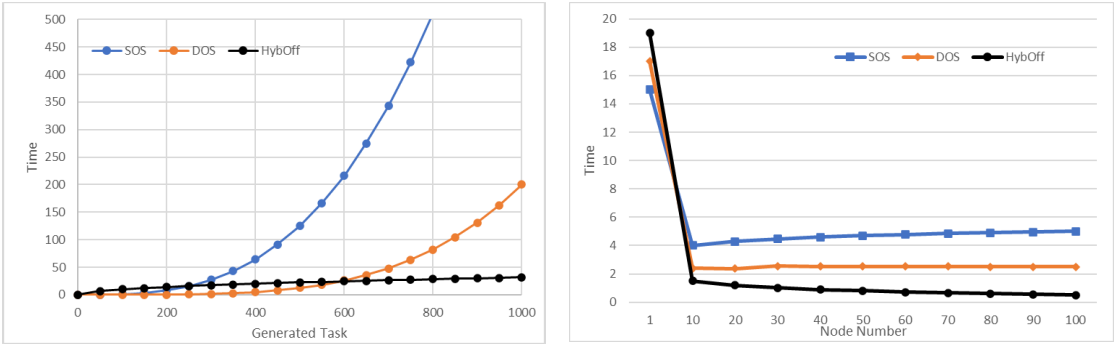
Figure 4.8: The percentage of healthy servers for the three algorithms with the same mean value but different standard deviations, where (a) is the percentage of balanced servers, and (b) is the standard deviation for algorithms.

However, standard deviation (σ) is crucial for assessing data dispersion. A smaller standard deviation signifies that data points are closely clustered around the central measure (Martinez and Bartholomew 2017). All algorithms were tailored to maintain equal load distribution among computing servers in this work. As previously defined, fog servers with computing loads within the 47.5% to 52.5% range are considered healthy. In this experiment, we tallied the number of servers loaded at approximately 50% for each class within each algorithm. Figure 4.8 (b) demonstrates that HybOff exhibited the lowest standard deviation, while SOS showed the highest. This indicates that HybOff had the most servers meeting the balanced criteria. Although DOS also upheld a substantial number of balanced servers, SOS struggled to keep servers within the target range. The performance results were 39%, 68%, and 95% for SOS, DOS, and HybOff, respectively. All three algorithms had the same mean value, $\overline{x}$=17. SOS, DOS, and HybOff had standard deviations of 20.4, 16.9, and 9.7, respectively. HybOff outperformed the other algorithms.

4.4.2.3. System Performance

It is essential to compare the performance of HybOff with SOS and DOS to assess the proposed algorithm's effectiveness. To evaluate each system's performance, we analyzed the execution of time-sensitive applications. As previously mentioned, all servers in the fog layer are tasked with serving time-sensitive applications locally without offloading. For resource-intensive applications, offloading is only considered when the computing servers are fully loaded (Xiao, Shu et al. 2022).

Figure 4.9 (a) depicts system execution time comparisons between HybOff, SOS, and DOS, evaluating their efficiency across various server scales and data sizes. HybOff exhibits notable effectiveness in handling time-sensitive applications and ensuring resource allocation in receiving servers. It excels in resource-intensive tasks by offloading to Complementary Servers (CoDs) without distant offloading, outperforming other algorithms. Figure 4.9 (b) illustrates the system execution time for the three algorithms with a fixed amount of generated data and an increasing number of fog servers. Initially, with just one fog server handling all the generated tasks, all the algorithms consumed significant time. However, as the number of fog servers increased, each algorithm exhibited a distinct behaviour. While all solutions showed a declining trend, HybOff consistently outperformed the others. With increasing servers, HybOff's performance led to reduced system execution time. Specifically, HybOff achieved a system performance 1.6 times and 1.4 times better than SOS and DOS, respectively, when operating with 100 servers. This demonstrates HybOff's efficiency in optimizing system performance and resource utilization as fog server numbers increase.



(a) (b)

Figure 4.9: TSAs Performance evaluation with (a) different data sizes and (b) different system scales.

4.5. Discussion

This study aimed to enhance our understanding of LB within FC environments by introducing a novel offloading algorithm called HybOff. HybOff was developed to address the inherent challenges associated with existing LB techniques. Our findings demonstrate that adopting a hybrid approach that combines the strengths of both static and dynamic algorithms significantly enhances system performance.

Key Findings: Our comprehensive analysis of the experimental results, as presented in Figures 7, 8, and 9, has revealed several key findings:

- **Decision Time:** HybOff's hybrid approach allows for direct offloading, eliminating the delay associated with decision-making in the offloading process. This contributes to faster and more efficient resource allocation.
- **Distance Offloading:** Unlike other algorithms that struggle with large-scale networks, HybOff excels by avoiding offloading to unknown fog locations, a characteristic more reflective of real-world FC scenarios. This reduces latency and improves system efficiency.
- **Decision Messages:** HybOff's static behaviour reduces the need for current system state messages, minimizing the exchange of messages among servers and reducing network bandwidth usage. This is crucial for optimizing network performance.
- **Superfluous Offloading:** While other algorithms may experience performance degradation when handling time-sensitive applications (TSAs), HybOff excels by keeping TSAs local, saving transmission time and network resources. It also efficiently manages heavy applications (HAs) by offloading them only to adjacent servers, thus minimizing network congestion.
- **Anti-Thrashing State:** HybOff effectively prevents the system from entering a thrashing state by employing the "disable offloading" function. This ensures that underloaded servers within each area share their resources with the most affected servers, ultimately optimizing system utilization.

These findings align with existing literature that underscores the effectiveness of dynamic offloading as a strategy for LB in FC. However, our study further demonstrates the viability of incorporating classical static offloading into modern network design. These results mark the first direct demonstration of this hybrid approach, offering valuable insights for future research in FC.

Limitations and Future Directions: Despite the promising findings, this study has identified two potential limitations:

- **High Load Scenarios:** HybOff may not operate efficiently in scenarios with a substantial load within a single cell. When all computing nodes in a cell reach their utilization limits, the "disabOff()" function activates, leading to offloading processes across cells or to the cloud, which may introduce undesired

consequences such as network congestion and distant offloading. Future research should explore sustainable solutions for high-load scenarios within a single cell.

- **Metric Selection:** While HybOff uses CPU load and network state as reference metrics to assess fog server loads, it does not consider other server metrics like memory usage and energy consumption. Future investigations could consider a more comprehensive set of metrics for a nuanced assessment.

Implications: These findings have both theoretical and practical implications. Reviving the use of static offloading techniques, previously deemed impractical in modern network design, emerges as a critical consideration. Adopting approaches like HybOff in industrial computing platforms may help reduce unnecessary network expansion and enhance system performance. In conclusion, HybOff offers a robust and efficient computing environment for fog systems, outperforming prevalent dynamic algorithms and providing valuable theoretical and practical insights for LB in FC scenarios. Future research can build on these insights to address the identified limitations and further advance the field of FC.

4.6. Stage 1c, Optimum LB Solution

This part of the research attempts to identify the optimum solution among task offloading and service placement models to improve the level of LB in the system after testing them with their peers in stages 1a and 1b. According to the real implementation of this research, Figure 1.5, this is stage 1c. It is related to research question 6, "*How could the performance of the proposed systems be evaluated in a fog environment in RQ4 and RQ.5 ?".* Whereas the central issue of this research is LB, resource utilization ratio and LB level are the main metrics that need to be evaluated according to (Kadhim and Seno 2018). Moreover, the number of hops is required to evaluate the impacts of the proposed models.

The comparison uses the fog architecture in Figure 2.2 to implement HybOff and RODSPP models. As clustering is the core solution in the HybOff model and cannot be avoided, a comparison must be held between two cells in both models. However, RODSPP does not have the cell concept. However, the shortest route towards the cloud is supposed to be a cell. Figure 4.10**Error! Reference source not found.**

shows the difference between cell construction in both models. To construct the experiment fairer, the cells in both models must have the same number of devices or computing power. While in RODSPP, we built a three-level of fog devices, we forced the HybOff model to create a cell with three devices by locating the other devices away. However, the formed cells have three devices in both models. In the experiment, all devices can receive the tasks directly from the IoT layer, even in the second two levels in RODSPP.

In HybOff, we did migrate the location of the services where all devices have all services, while RODSPP is changed according to its configuration. The experiment uses the exact configurations for both models, as shown in Table 3.3, except no gateway is used. It follows procedures in (Sulimani, Sajjad et al. 2022) and subsection 3.4.1 using iFogSim.

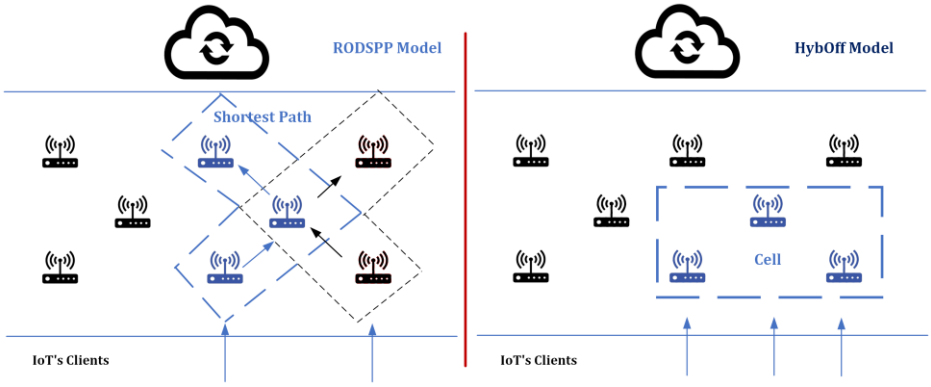


Figure 4.10: The definition of cell in models.

Resource utilization (CPU usage) is the first evaluation metric to be tested; this research considers CPU as a core resource while ignored the others as discussed in subsection Figure 4.11. Equation 4.9 calculates the utilization level in the cell system for both models.

All task offloading processes are initiated by the father to the following two levels. Figure 4.10 depicts that there is a possibility that two fathers might use the same intermediate devices. On the other hand, in HybOff, all devices work independently and offload to their CPDs; device one cooperates with device three in this experiment while device two works independently. Because this study is more concerned with sensitive applications, we have generated more light (time-sensitive) than heavy applications by twenty percent.

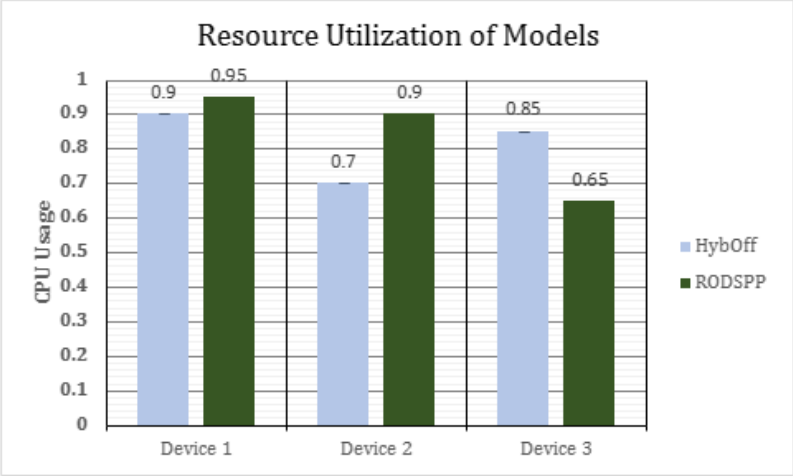


Figure 4.11: CPU usage of the devices about their topology distribution. Experiment with two applications.

Secondly, the number of hops is the second metric in this comparison. It mainly relies on the number of offloading processes and changes the location of cloud services among the levels; the increasing number of hops impacts the network state, as discussed in subsection 3.5. In the RODSPP model, the offloading processes to execute the task are considered a hop. For example, if one arrival task is divided into interrelated instances, some instances must be offloaded to the following levels for execution. On the other hand, in HybOff, it is required to offload to its CPD.

However, as we configured both models, Figure 4.12, by giving an advantage for time-sensitive over heavy applications to be executed by the edge device, an increasing number of heavily generated tasks is required to force the system to offload. Although the sensitive applications are kept at the bottom level of generation per device, we setup the heavy tasks to be random.

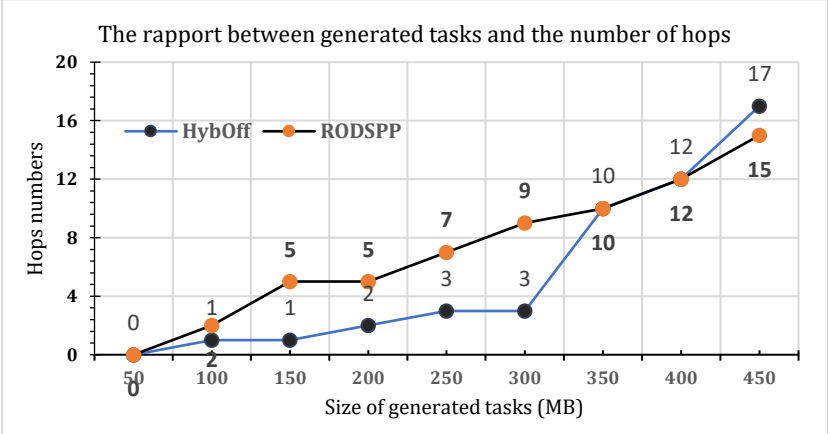


Figure 4.12: Hops count with different situations. Experiment with two applications, two users, and three fog levels: (a) changing the number of generated tasks.

The last metric in this stage (1c) is the level of LB. It is a good indicator that all computing units have a fair workload regardless utilization level; consequently, the success of the tested model. Equation 4.13 was used to calculate the level in both models. In this experiment, as shown in Figure 4.13, we increased the number of devices to measure the level of LB for the whole system.

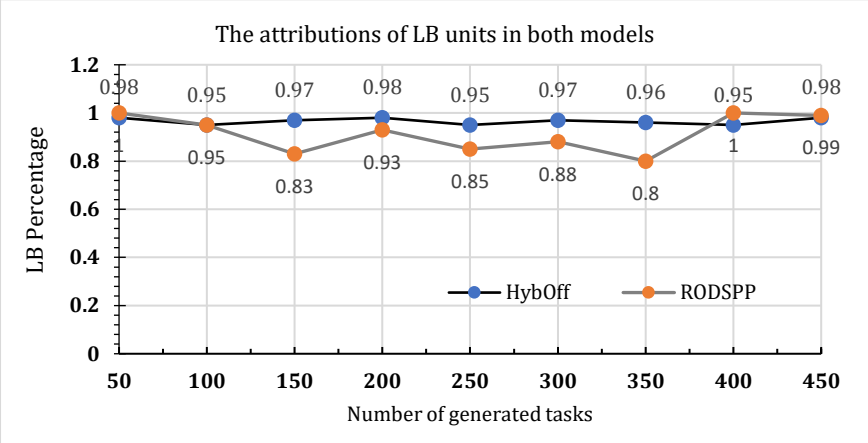


Figure 4.13: The level of LB for the proposed models under different size of generated tasks and constant computing power.

4.6.1. DISCUSSION

The series labelled HybOff and RODSPP in Figure 4.11 were thus analysed. The chart shows that device one in RODSPP loaded more than HybOff. It is clear that the father consumes its resources to execute a high number of sensitive applications; the heavy tasks is offloaded to other levels, to device two and three. In HybOff, devices one and three execute the sensitive applications locally and assign offloading processes for heavy tasks. In summary, both models have the same performance in utilising their resources.

Figure 4.11 illustrates the impacts of receiving many tasks between models. Generally, the number of hops is proportional to the number of heavy tasks, while the sensitive applications are at a low generation rate. In the experiment, HybOff maintains the number of hops at the low compared to the other model. The reason behind that is the HybOff offload to the un-heavy device while in RODSPP, it offloads to the next device in the shortest route, which might be used by another shortest route. In this case, RODSPP will offload to the third level and then to the cloud (in our setting). However, HybOff begins increasing the number of hops receiving 400MB of generated data; it seems that HybOff loses its features when the receiving workload exceeds the computing powers of cooperated pairs. However, both

models show identical performance between 350 and 400 MB. Even though HybOff outdoes RODSPP, it is mandated to invent a comprehensive solution for HybOff to confound this situation.

Figure 4.12 shows the performance of both models to balance the load. Even though RODSPP records a high level of LB, it is clear that it cannot maintain its performance, and there is a fluctuation. This behaviour is due to the device in the third level not being used and waiting for workloads. On the other hand, HybOff maintains its performance by keeping all devices have the same load. The cooperative opposing-load philosophy is behind this performance.

In summary, the HybOff approach can secure a stable and robust computing environment for fog systems. It achieves outstanding performance compared to the RODSPP models.

4.6.2. LIMITATIONS

This subsection is concerning to answer the seventh research question, named: "What are the limitations of the optimum LB solution in this research?". However, there are at least two potential limitations concerning the results of this study. The first limitation concerns with a massive load in one cell. A second potential limitation is using CPU utilization as the only considered metric while ignoring other metrics.

One limitation of this study is that if one cell faces a high amount of arrival tasks and they are fully loaded, HybOff achieves unexpected outcomes, in this case, there is no benefit in Hybrid offloading. It is possible that all computing nodes reach to the utilization limit which allow the disabOff( ) function to block the offloading feature inside the cell. Consequentiality, HybOff, in this case, will start the offloading process across cells or the cloud to find free resources which might be located a distance away; in this scenario, HybOff will follow prevalent offloading, which, as mentioned, has drawbacks. It generates the undesirable phenomena, such as network congested and distant offloading. Accordingly, HybOff cannot work efficiently within overloaded cells.

While HybOff uses the load of the CPU and network state as references to evaluate the load of fog devices, it ignores other metrics in the device, such as

memory usage and energy consumption. However, the authors believe that the CPU is the core of any computing system; it is considered a primary metric in the evaluation process.

To address the first limitation, we need to find a solution with a sustainable concept that might be suitable for future work. Although the present research cannot rule out this limitation, it seems useful to point out issues that may conflict with the proposed model.

Despite this limitation, these results suggest several theoretical and practical implications. For example, igniting the static offloading technique is critically essential, which programmers have avoided using in the past, as it is considered a non-practical strategy in network design. Moreover, Industrial computing platforms can improve their system performance by following this approach, which attempts to prevent superfluous network expansion.

4.7. CONCLUSION.

This work aimed to enhance the performance of critical applications in large-scale fog networks by introducing a novel algorithm named "HybOff". HybOff represents a LB offloading technique that adeptly harnesses the benefits of both static and dynamic offloading methods, resulting in substantial performance improvements for time-sensitive applications, regardless of network scale. The offloading strategies generated by each algorithm in this investigation were simulated utilizing the iFogSim platform. Through a comparative analysis of diverse metrics encompassing resource utilization, load distribution, and system performance, we discerned the merits and demerits of each approach. These algorithms' outcomes affirm that regardless of network size, HybOff consistently fulfils the requisites of Application Service Dependencies (ASD).

Furthermore, the experimental results strongly corroborate the efficacy of HybOff. It demonstrates a notable reduction in the volume of offloading messages, distance traversed, and the repercussions of offloading decisions. These outcomes effectively mitigate the inherent deficiencies encountered in traditional offloading techniques. Notably, the proposed algorithm enhances LB by an impressive 97%, a substantial improvement compared to the 64% and 88% achieved by SOS and DOS, respectively. Moreover, it elevates the average system utilization rate by 50% and enhances system performance by 1.6 times and 1.4 times compared to SOS and DOS, respectively.

5. CHAPTER, A COMPREHENSIVE SOLUTION: A SUSTAINABLE LOAD BALANCING MONITORING SYSTEM FOR THE HIDDEN COSTS IN FOG COMPUTING

Abstract: Due to the complex behaviour in the LB process, prevalent offloading techniques in fog computing incur hidden costs, such as network congestion, distance offloading, and decision time. Earlier, in our previous work, we introduced the Hybrid Offloading Approach (HybOff) to improve the performance of time-sensitive applications for large-scale fog networks, which marked the first attempt to mitigate these costs. Although HybOff is designed to address these issues, there are specific cases where it loses its effectiveness and reverts to the behaviour of prevalent techniques. This article presents a SlbmS to provide a comprehensive solution for the HybOff model. It has struggled with the increased demand for data processing in highly loaded cells and their limited physical resources. SlbmS is a tool that collaborates with HybOff using machine learning to restore its features if it implements SlbmS's recommendations. According to the framework, SlbmS is constructed in two sequential stages to achieve the research objective. The investigation assesses the performance of HybOff with and without SlbmS in congested cells. The simulation experiment demonstrates that implementing the proposed recommendation of quality or quantity results in HybOff regaining its features. Adequate planning for the expansion process enables HybOff to overcome the challenges posed by exceptional cases. This work proves that offloading is an alternative solution for resource provisioning.

5.1. INTRODUCTION

In the internet era, the central computing concept has become so vital that we can no longer revert to primitive, decentralised systems. On the ground, cloud computing is the practical implementation of central computing infrastructure. Any business must establish direct connections and communication with its intelligent products to manage its operations. For this purpose, the cloud computing model has been widely recognised as the appropriate infrastructure for efficiently running and managing widely distributed IoT (Internet of Things) products. IoT is a telecommunication system that enables many processing devices (things) to exchange data over a public network, thus allowing businesses to operate with minimal human intervention (Catruc and Iosifescu 2020). It is considered a fundamental tool that facilitates the interaction between cloud computing and the environment. Most of the surrounding intelligent products, which overcome daily challenges, rely on IoT technology, implying that they may not operate or function fully without an internet connection (Madakam, Lake et al. 2015). This setup allows IoT technology to increase widely and its data to grow gradually, enhancing cloud systems' popularity while influencing public networks' efficiency.

On the other hand, when the public network slows down, time-sensitive applications are the ones that suffer the most. IoT applications that rely on timing must be served within a reasonable time limit for them to perform well. It's important to note that all IoT applications utilise the same public network, designed to serve all applications impartially (Wójcicki, Biegańska et al. 2022). Consequently, providers of time-sensitive applications must consider this when introducing critical services, such as e-health, smart grids, and unmanned vehicles (Goel, Abeni et al. 2002, Lu, Wu et al. 2023).

Unfortunately, cloud computing cannot offer the required level of service for these types of applications due to the unpredictable performance of public networks (Ferrer, Marquès et al. 2019). As extensively discussed in prior research, creating a well-balanced fog system is crucial to overcoming this situation (Chandak and Ray 2019). However, this work considers offloading the appropriate technique to enhance LB in FC, enabling cloud technology to continue functioning effectively for

time-sensitive applications. Therefore, providing a robust solution for offloading is essential.

Conceptually, once LB systems start offloading, local fog resources become exhausted, and the workload significantly increases. Effective resource provisioning strategies are crucial to address a situation characterised by resource shortages. Most existing resource provisioning research in FC employs granularity services such as virtual resource allocation, container-based provisioning, and task allocation to enhance the capabilities of available local resources (Shakarami, Shakarami et al. 2022). However, these solutions are constrained by physical resources and do not expand beyond that, as it has not been explored previously. Therefore, this research aims to explore this new research dimension in resource provisioning.

Considering the gap in this research, this work needs to incorporate the concept of to assess the validity of this new dimension. Sustainability in Information Technology (IT) fields generally refers to keeping the system running efficiently with minimal changes. Due to the proposed avoidance of expanding physical resources, however, all prevalent offloading solutions are incongruent with this concept. Hence, the solution explored in this study aims to develop a long-term-based solution (Gonzalez-Mejía, Eason et al. 2012).

To achieve this, we need to refine resource provisioning in the offloading model by studying fog devices' quality and quantity of coverage (Lu, Wu et al. 2023). Quality of resource provisioning entails the system reorganising the range of underutilised fog devices to assist nearby overloaded devices, considering each fog device has a limited geographical area to cover (Hermann, Emmelmann et al. 2007). On the other hand, the quantity of resource provisioning involves altering the physical number of fog devices. This work primarily focuses on offloading, which can be considered an alternative technique for resource provisioning. If we succeed in this study by increasing local resources at the lowest cost, we will regard offloading as a viable alternative to resource provisioning in a fog environment.

The primary objective of this research is to address the challenges faced by the Hybrid Offloading (HybOff) model, which have arisen due to the high demand for

data processing in already congested cells and the limited physical fog resources. This inevitable scenario compels HybOff to adopt the behaviour of prevalent offloading solutions. While existing research has consistently advocated for offloading as a critical solution (Aazam, Zeadally et al. 2018, He, Lu et al. 2020, Dai, Xiao et al. 2022, Chakraborty and Mazumdar 2023), this work, in contrast, aims to minimise or even eliminate offloading processes and their associated consequences (Lin, Kar et al. 2023). To achieve this, we propose a novel approach involving minimal changes to the network topology using genetic algorithms and reinforcement learning (RL) (Ray 2019). As far as our knowledge extends, this research problem remains unresolved, and we aim to provide a solution in this work. Importantly, our approach may also resolve the challenges prevalent offloading techniques pose.

- Key contributions of this paper include:
- The introduction of a new method for creating a sustainable LB system.
- A comprehensive discussion of the hidden costs involved in ensuring the performance of HybOff as an innovative solution.
- A demonstration of how the concept of sustainability can be applied to the proposed model.
- Presentation of the results obtained from the comparative experiments conducted in this study.

The subsequent sections of this manuscript are structured as follows: Section 2 provides background information on the research area. Section 3 outlines the research problem. In Section 4, we describe the proposed sustainable load-balancing monitoring system. Section 5 details the implementation and evaluation aspects, while Section 6 discusses the results. Finally, Section 7 presents the conclusion and outlines avenues for future research.

5.2. BACKGROUND OF THE RESEARCH AREA

In this section, we provide a brief introduction to sustainability, LB, and Machine Learning (ML).

5.2.1. SUSTAINABILITY

In the General Conference of the United Nations in 1984, the World Commission on Environmental and Development (WCED) introduced the term “sustainability” with

a broad meaning (Gonzalez-Mejía, Eason et al. 2012). The concept of sustainability now extends across various fields, including agriculture, manufacturing, health, and information technology (IT). While numerous definitions of sustainability exist in the computing field, they all share a common core idea. For instance, Gonzalez-Mejía, Eason et al. (2012) define sustainability in IT as "maintaining the functionality of a system without experiencing significant changes in its conditions over time." However, to apply sustainability in this research, we need to introduce the concept of system utilisation. Utilisation is a process that not only maximises the usage of available resources but also minimises waste to keep the system fully equipped (Kadhim and Seno 2018). Therefore, we can redefine sustainability as the process aimed at maintaining the functionality of a system at the highest level of utilisation, thus increasing the benefits of system components during their limited lifetimes without significant changes in their conditions over time. This research adopts this definition to address the current research dilemma.

5.2.2. Reinforcement Learning (RL)

Machine Learning (ML) is a process that enables a computer program to improve its performance on specific tasks by learning from past experiences. ML algorithms are generally classified into three main categories: supervised, unsupervised, and Reinforcement Learning (RL) (Kaelbling, Littman et al. 1996). Supervised learning involves training the system using labelled data to predict continuous values. Unsupervised learning focuses on labelling or clustering based on similarities. In real-life scenarios, systems acquire knowledge by testing all possible action combinations before identifying the optimal solution. RL is the challenge an agent faces that learns behaviour through trial-and-error interactions with a dynamic environment. It comprises three key elements: agents, actions, and rewards, collectively determining the policy. The policy is developed as the agent interacts with the environment, earning rewards while striving to achieve a specific task (Prudencio, Maximo et al. 2023). Therefore, the selected policy represents a sequence of actions that lead to the endpoint with the fewest rewards.

For instance, consider a scenario where you are tasked with escaping from an escape room using a series of moves (actions). At each step, you have three activities to choose from: forward, left, or right. Hitting an obstacle forces you to restart, with

each successful action yielding rewards. If you start moving and after x actions hit an obstacle, you collect x points. On the next attempt, the agent chooses a different activity and, after y steps, hits an obstacle, earning y points (y > x). The system continues to explore the environment until it finds the optimal policy. Consequently, the selected policy should guide the agent to reach the endpoint via the shortest path.

5.2.3. Load Balancing in a Fog Environment

In FC, LB is a mechanism that enhances the efficiency of resource allocation by determining the optimal device to receive streaming tasks. Its goal is to distribute incoming tasks evenly to equip all active computing units fully. Each LB algorithm employs a distinct mechanism to achieve this objective. Generally, when managed effectively, LB is utilised to increase system throughput, resource utilisation, performance, and reliability. Although LB research has inspired various solutions to optimise resource allocation, the explosive growth of IoT data, users, and application requirements has significantly burdened the fog layer, which possesses limited resources.

5.2.4. Related Works

Most LB schemes typically acknowledge hidden costs as an inherent factor that cannot be avoided. They aim to minimise the adverse effects of these costs. However, in this section, we delve into LB-related works attempting to circumvent these costs or significantly mitigate their impact through sustainable research solutions.

Puthal, Obaidat et al. (2018) propose an innovative LB technique to authenticate Edge Data Centres (EDCs) and identify less loaded EDCs for task allocation. Their process employs the Breadth First Search method to explore neighbouring EDCs, enabling overloaded EDCs to find optimal destinations for offloading. However, this work introduces a bi-objective approach where cooperation among EDCs involves an identification process that can lead to cooperation termination if it fails. While this paper includes sustainability in its title, it does not provide a long-term resolution for the ongoing challenge of numerous offloading processes among EDCs.

Abedin, Bairagi et al. (2018) introduce an LB algorithm designed to redistribute tasks among edge centres in metropolitan networks, focusing on reducing average

response times, especially in wireless networks. Rafique, Shah et al. (2019) propose a bio-inspired hybrid algorithm (NBIHA) for LB in fog environments. They reduce average energy consumption and response times by employing efficient task scheduling. NBIHA operates meta-heuristic particle swarm optimisation (MPSO) as a scheduler within the fog layer. Simulations using iFogSim demonstrate favourable outcomes in terms of resource management.

Arshad, Khattak et al. (2018) analyse nature-based algorithms, precisely the binary bat algorithm (BBA) and pigeon-inspired optimisation (PIO), within a fog architecture comprising smart homes in the first layer and cloudlets in the second layer. The study evaluates the energy consumption of the cloudlets. (Javaid, Butt et al. 2019) propose a nature-inspired Cuckoo search algorithm combined with flower pollination and levy walk distribution to enhance processing and response times while reducing microgrid and data transfer costs. Lu, Wu et al. (2023) investigated a multi-user mobile edge computing architecture for high-reliability and low-latency scenarios, analysing the task queue using extreme value theory. Singh (2022)designed an LB system based on fuzzy logic with various levels of fuzzy control design and tuning for analysing links and managing traffic.

In the context of fog/cloud platforms, Maswood, Rahman et al. (2020) introduce a mixed-integer linear programming (MILP) LB model. This model aims to optimise network bandwidth and server resource utilisation, evaluating the model holistically at both the network and server levels.

5.3. Research Problem

FC is strategically designed to deliver instantaneous services near clients, ensuring the continuous provision of critical cloud services. FC has evolved in various dimensions to meet the demands of time-sensitive applications and accommodate the recent surge in IoT devices and users. However, a pressing need remains for enhancing LB techniques explicitly tailored to fog environments to optimise the utilisation of fog resources as clients deplete ground resources and the burden on devices escalates.

Recent research underscores that offloading is pivotal to LB in distributed environments. Numerous studies have introduced various offloading approaches,

including dynamic, age-based, and hybrid strategies, to enhance the efficiency of fog environments (Hussein and Mousa 2020, Liu, Xu et al. 2020, Dai, Xiao et al. 2022, Sheikh Sofla, Haghi Kashani et al. 2022, Tran-Dang and Kim 2023).

In our prior work, we introduced the HybOff algorithm as a novel solution to address certain inherent limitations of static and dynamic offloading approaches, such as reliability issues and high communication latency associated with static and dynamic offloading methods. In this research, we develop a sustainable solution that operates with HybOff. The goal is to mitigate the weaknesses of the HybOff algorithm and restore its key features to ensure robust and efficient offloading in fog environments.

5.3.1. The Prevalent Solution

According to our observation, we have noticed that all the prevalent offloading approaches use the present system state theme, in which the heavy devices read the environment (gathering attributes) and give an offloading action to redirect the excess tasks to the target device. This is repeated several times when there is a necessity for extra resources. This theme generates a high volume of exchanged messages with the peer devices; we can call it decision messages. They seek to explore unused resources to cover the shortage in the affected areas without the intention to increase the number of served devices (Sheikh Sofla, Haghi Kashani et al. 2022). Although all research approaches seek to enhance offloading strategy, they ignore these hidden costs.

As illustrated in Figure 5.1, an infinite series of interaction and offloading processes among devices will occur due to the unchanging quantity or quality of physical resources in the field. These processes make the network situation seem to get worse over time. Keeping the networks congested will escalate the latency in the system (Kuempel, Adams et al. 2018), which is a decline in the main objective of FC.

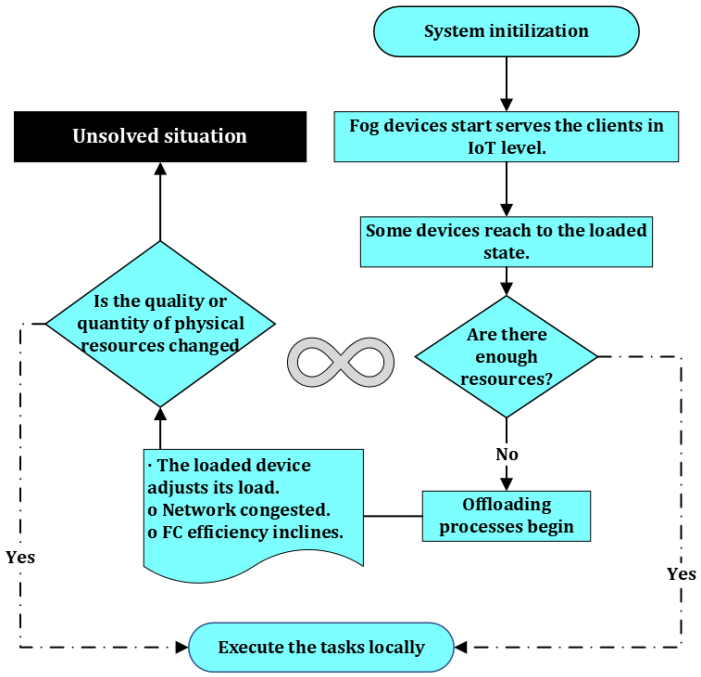


Figure 5.1: Prevalent Offloading Process Flowchart.

While HybOff is designed to address the challenges in FC, there are specific scenarios where it tends to adopt the behaviour of prevalent algorithms, particularly when confronted with heavy loads in particular cells. In such cases, HybOff resorts to "distance offloading," sending tasks to other cells or the cloud, increasing system latency. This research assumes that HybOff operates efficiently in maintaining balanced and fully utilised computing nodes, except under these circumstances. Therefore, without improving the physical resources in these affected areas, HybOff may have long-term adverse consequences, diminishing the effectiveness of software solutions.

Given this scenario and following the flowchart, enhancing the quantity of physical resources emerges as the most viable approach to breaking this endless loop. This step aims to minimise or halt the constant message exchanges. Consequently, expanding the network, either partially or entirely, appears to be the logical solution to support the affected areas. However, this decision may not be suitable if there is uncertainty about the full utilisation of all fog devices, especially considering the varying efficiency of LB algorithms.

In such cases, network expansion can lead to a unique network state, which we propose as "network oversizing." This state arises when the network expands without adequate planning. Excess resources and power are wasted when the

provided resources surpass the demand. Network oversizing is particularly prevalent in smaller businesses that may not conduct thorough network studies or planning before expanding. Importantly, network oversizing contradicts the sustainability definition adopted in this research, which seeks to avoid significant or unwarranted physical changes.

Therefore, engaging in meticulous planning for the expansion process is imperative, including estimating the number of Access Points (APs) required before actual implementation. Adequate planning ensures that the performance of FC remains at an acceptable level without incurring excessive costs.

5.3.2. EXPLORING THE NEW DIRECTION OF RESEARCH

To tackle the dilemma effectively, it is imperative to implement an intelligent and autonomous monitoring system capable of monitoring, evaluating, and proposing solutions to enhance the resource management system. The proposed solution should bridge the gaps left by existing solutions, addressing issues such as decision messaging, network congestion, and network oversizing. The primary focus of the proposed solution should be to augment the computing power available on the ground through meticulous resource planning, as depicted in Figure 5.2.

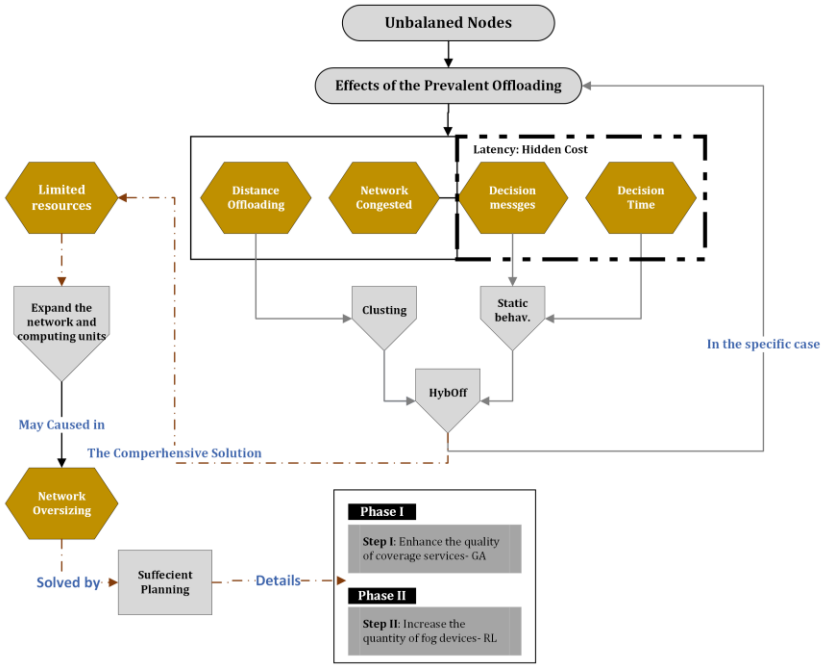


Figure 5.2: Possible Solutions for the Problem of Research.

Hence, this is the opportune moment to create a comprehensive solution that allows for physical modifications (both in terms of quality and quantity) in the

network topology. Constructing this proposed system will unfold in two sequential phases. The initial phase aims to enhance the quality of coverage provided by computing devices or Access Points (APs). In contrast, the subsequent phase seeks to augment the quantity of computing devices or APs. Both phases are designed to bolster physical resources.

In the first phase, the proposed model will continuously monitor the load readings of fog devices and provide recommendations for altering the network topology based on analysing the system's load data. In this phase, the proposed model will utilise genetic algorithms to study, analyse, and simulate (offline) all feasible recommendations before selecting and implementing the most suitable one. In this phase, a critical piece of advice is repositioning fog devices closer to overloaded areas, thereby enhancing coverage quality in those affected regions.

However, suppose the simulation results for rearrangement (quality) are deemed insufficient. In that case, the model will transition to the second phase, involving the evaluation of the impact of adding a certain number of APs (quantity). This step constitutes a partial or complete network expansion within the overloaded areas if deemed necessary. This approach provides overloaded devices with additional physical resources, resulting in a natural reduction or cessation of offloading processes. This, in turn, alleviates the strain on network bandwidth and puts an end to the cycle of infinite processes. As a result, the concept of sustainability is effectively preserved and realised.

5.4. The Proposed Sustainable Load-Balancing Monitoring System (SlbmS)

This section introduces the architecture and structure of the proposed sustainable solution, SlbmS. It is not an LB solution but a recommendation system to facilitate sufficient planning for enhancing local resources. SlbmS heavily relies on the HybOff algorithm, as depicted in Figure 5.3, where HybOff represents the endorsed LB algorithm in this research due to its exceptional resource utilisation capabilities.

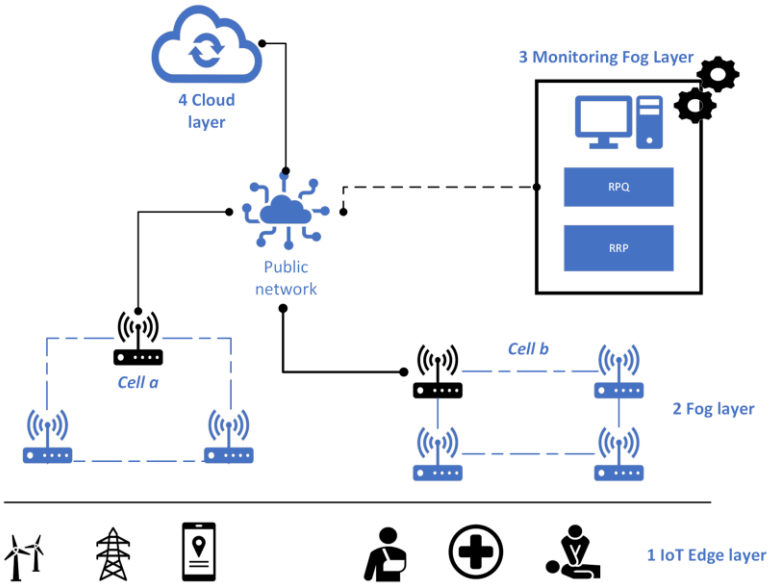


Figure 5.3: The Proposed Model Architecture.

Time-sensitive applications demand a highly reliable and continuously available FC system. This research predominantly investigates such applications, making SlbmS potentially applicable to a wide range of critical fog environments. Consequently, this research is focused on designing a robust IoT fog system primarily geared toward serving essential applications.

The outcome of this research stage comprises a set of recommendations for implementing physical changes in the network topology. These recommendations should be executed by repositioning fog devices or Access Points (APs). In this work, we delve into the theoretical aspects of sustainable theory. SlbmS is subjected to testing both before and after these changes through simulation.

5.4.1. The Architecture

The proposed model's architecture, as illustrated in Figure 5.4, comprises four distinct layers: (1) IoT Edge Layer, (2) Fog Layer, (3) Monitoring Fog Layer (MFL), and (4) Cloud Data Centre Layer. Each layer serves a specific purpose within the system.

- **IoT Edge Layer:** This layer connects IoT objects with limited processing capabilities, including mobile phones, laptops, actuators, and sensors, to the fog system. It facilitates interaction between these IoT devices and the fog environment.

- **Fog Layer:** The fog layer is designed to manage the influx of user requests and data, processing them locally or redirecting them to other fog devices or the cloud. It is subdivided into cells based on relative distances, following the HybOff model's proposal. Additionally, this layer houses the LB algorithm, which plays a crucial role in optimising resource allocation.

- **Monitoring Fog Layer (MFL):** The third layer, MFL, has direct access to the fog devices. Within MFL, the SlbmS model is situated. SlbmS is responsible for enhancing HybOff's performance and devising long-term solutions to LB challenges. Significantly, MFL's absence does not disrupt the fog system's operation since it is not involved in data processing. This layer seamlessly complements the HybOff architecture.

- **Cloud Data Centre Layer:** The cloud layer manages data exchanges with the fog layer while fulfilling its primary responsibilities. It encompasses many high-performance servers that deliver stable and reliable services.

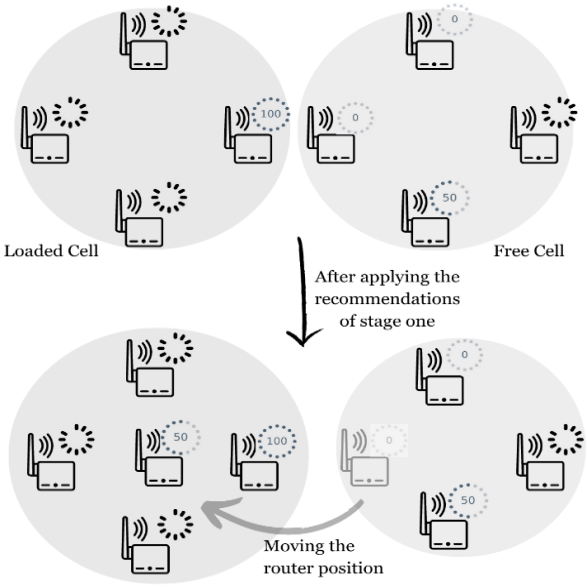


Figure 5.4: The Effect of RPQ-Stage One.

The organisation responsible for the FC system's administration oversees the operation and maintenance of devices in the second and third layers. Conversely, a third-party organisation, such as Google or Amazon, owns the fourth layer, which provides on-demand cloud services.

5.4.2. The Structure Of The Model

In this section, SlbmS is introduced and located within the MFL. SlbmS is a remote system that establishes direct connections with the master fog devices in the fog layer. SlbmS comprises a Resource Provisioning Quality module (RPQ) and a Recommendation System for Resource Provisioning (RRP). However, the following subsections will detail the construction of these systems.

5.4.2.1. Resource Provisioning Quality System (RPQ)

RPQ is a software module aimed at reorganising the distribution of Access Points (APs) in the fog layer to provide additional computing resources in highly affected areas. It represents the first stage of SlbmS. RPQ operates through the following steps when called:

- **Data Collection:** RPQ gathers the necessary attributes to build a background understanding of device loads.
- **Data Preparation:** Using a genetic algorithm, it prepares the required data to identify extra resources near the overloaded APs.
- **Evaluation:** The system evaluates the effectiveness of these changes.
- **Recommendations:** Finally, the system sends suggestions to be applied. Once this phase is completed, the system remains inactive until requested again.
- **RPQ comprises two sub-modules:** the Classifying Cells and Devices Module (CrM) and the Arrangement and Simulation Module (ASM). Figure 5.4 illustrates the benefits of the first phase.
- **Classifying Module (CrM):** This module's primary objective is to prepare the necessary data from the IoT layer. CrM is responsible for executing the first two steps of RPQ, involving the classification process by collecting the required attributes of the fog devices. CrM also contains additional features not used in HybOff.

Various evaluation metrics are employed to study and assess the proposed model. According to HybOff, devices are classified as either heavy or light. RPQ seeks to reclassify fog devices into overloaded, balanced, and underloaded categories. The core goal of this research is to address the main weaknesses of HybOff, such as network congestion and distant overloading. Therefore, more straightforward metrics focusing on network traffic and computing unit load are preferred. Studying the number of incoming and migrated tasks is a suitable indicator for classification.

These attributes are stored in the Device Attributes Table (DAT) in MFL, as shown in Figure 5.5. To represent the attributes of fog devices as described, you can use the following notation:

a) Average Arrival Tasks from the IoT Layer:
Symbol: $(\boldsymbol{AT_i^{\varphi_{IoT}}})$**,**
Description: The average number of tasks per second from the IoT layer to fog device i.

b) Arrival Tasks from Paired Devices:
Symbol: $(AT_i^{\varphi_{OP}})$,
Description: The number of tasks per second from paired devices to fog devices i.

c) Offloading Tasks:
Symbol: (OT_i^{φ}),
Description: The number of tasks per second that need to be offloaded by fog device i.

d) Distance Offloading Tasks:
Symbol: $(\boldsymbol{OT_i^{\varphi_{DO}}})$,
Description: The number of tasks per second that require distance offloading involving other fog devices or the cloud by fog device i.

e) Device Capacity in Data Processing per Second:
Symbol: Fn_i^{Cap}
Description: The capacity of arrival tasks to fog device i in terms of tasks processing per second.

These attributes can evaluate fog devices and make decisions in a fog environment. These attributes are utilised for AP evaluation.

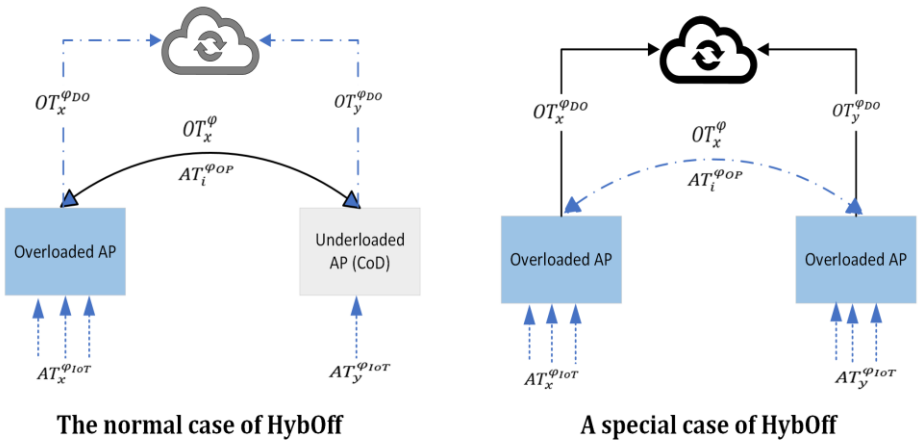


Figure 5.5: Type of exchanged data in APs.

On the other hand, the Adapted Weight (AW) serves as a gain factor in the evaluation process. It confirms that fog devices are eager to increase the number of

free resources in the cell and identifies the least engaged device (τ). Equation 1 outlines the formula used to calculate the AW for each device in the fog layer:

$$\mathrm{AW_i} = \beta \times \frac{AT_i^{\varphi_{IoT}} + AT_i^{\varphi_{OP}} + Fn_i^{Cap}}{OT_i^{\varphi} + OT_y^{\varphi_{DO}}} \quad , \tag{5.1}$$

where AW is adjusted based on the value of β , used to ensure fair evaluation of APs, considering offloading processes that collectively reduce the load in the cell. The subsequent submodule generates updated values of β. After this step, the data is ready for the classification process.

- *Arrangement and simulation module (ASM)*

ASM is a software submodule of RPQ that leverages the calculations made by CrM to make informed decisions after thorough planning. ASM is responsible for carrying out the second two steps of RPQ. It employs a genetic algorithm (GA) to determine the optimal value of β to enhance the AW for each device. Algorithm 1, OBS, outlines the overall steps of the GA for the optimisation process, which aims to find the ideal value of β. The GA's population comprises all fog devices in the system and a set of virtual tasks. This step leads to the classification of fog devices, as depicted in Table 5.1.

ALGORITHM 1: Optimal Beta Setting (OBS)

Output: Optimised β ($\beta_\circ$)
Input: Set of available FSs, Set of assumed tasks, Initial values for β
1. Generate a new random population of chromosomes for pair (β).
2. For each new population do
3. For each chromosome containing (β) do
4. Assign tasks and calculate LB level (LBL).
5. Next
6. Do mutation: (i) Select a random particle. (ii) Change its value.
7. Do crossover: (i) Select two random particles. (ii) Exchange them. (iii) Calculate their fitness. (iv) Chose the best of them
8. Calculate the best values for (β) with the highest LBL.
9. Next

After finding the AW for each AP, the ASM determines the Average AW (AAW) and the Threshold AW (TAW) using Equation 5.2.

$$TAW = Fn_{AWT}^{i} | (\gamma_i \leq 10\%), \tag{5.2}$$

where γ_i is the average of the arrival tasks in the cell. The ASM is responsible for periodically updating the attributes of APs to classify them as (1) balanced, (2) overloaded, or (3) underloaded, according to Table 5.1 and the accompanying rules. Note that the capacity of Fog devices (AP) is 50 Tsks/s.

- AP ==> Underloaded if AW >= TAW.
- AP ==> Balanced if TAW > AW >= AAW.
- AP ==> Overloaded if AW < TAW.

Table 5.1: DEVICES ATTRIBUTES TABLE (DAT).

Fn_m	$Cell$	$AT_i^{\varphi_{IoT}}$	$AT_i^{\varphi_{OP}}$	OT_i^{φ}	$OT_y^{\varphi_{DO}}$	AW_i	Status
Fn_1	3	150 Tsk/s	0 Tsk/s	3 Tsk/s	70 Tsk/s	0.2	OL
Fn_3	3	180 Tsk/s	0 Tsk/s	5 Tsk/s	98 Tsk/s	0.15	OL
Fn_4	3	192 Tsk/s	0 Tsk/s	3 Tsk/s	122 Tsk/s	0.14	OL
Fn_2	2	4 Tsk/s	0 Tsk/s	0 Tsk/s	1 Tsk/s	3.78	UL
Fn_1	2	50 Tsk/s	0 Tsk/s	4 Tsk/s	1 Tsk/s	1.4	UL
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

On the other hand, to classify the cells in this model, the most affected cell (MaC) is calculated. It is the metric that represents the status of system cells; MaC is the most important metric to find out the affected area and increase resources there. Equation 5.3 shows how it should be calculated, where n is the number of APs in the cell:

$$Ce_i^{MaC} = \frac{\gamma_i}{\sum_{i=1}^{n} Fn_i^{Cap}} \tag{5.3}$$

ASM's outcome is finding the set of APs to be moved into overloaded cells. The relocation cost of APs is ignored in this study. Despite the drawbacks of the relocation process for some clients, it will improve service in other congested network parts. The ASM sends the recommendation, and the system administrator will decide whether to reposition them; if there is no recommendation, RPQ will hand over the overloaded cells to RRP to increase the quantity of APs in the system.

5.4.2.2. RECOMMENDATIONS SYSTEM OF RESOURCE PROVISIONING (RRP)

The recommendations system for resource provisioning (RRP) is the second phase of the comprehensive solution. The RRP has an agent: RL-based resource provisioning (RP). RRP utilises a modified weight parameter. It prioritises the available APs in congested cells based on the AW value to be enhanced. The priority is given to the cell with the highest AW. The goal of the RRP module is to estimate τ which represents the number of added APs to the specific cell. τ has a minimum arrival task and offloading process ratio value with sufficient moving resources. τ is the less effective device on the network coverage that gains the highest score of AW, such as Fn_2 in cell two, as shown in Table 1. The following equations show the formula that governs this calculation:

$$\tau_i = n_i - \frac{\gamma_i}{\alpha \times Fn_i^{Cap}} \tag{5.4}$$

where α is the minimum reserve computing power. The system considers the cell as one block of computing cells. We ignore the exact location. As shown in Table 2, suppose cell three has a MaC of 3.3 and cell two has a MaC of 0.06; they are classified as overloaded and underloaded, respectively. In this case, the RRP tests all combinations to add nine APs to cell 3, where τ equals nine, while cell two can move out one AP to cell three. Despite the high number of APs recommended, this capacity gives space for future loads. Figure 5.6 depicts the overarching stages of the proposed SlbmS.

Table 5.2: STATES OF THE CELLS OF THE SYSTEM.

i	n	Ce_i^{MaC}	τ	Status
1	3	2.17	7	B
2	2	0.06	-1	UL
3	5	3.3	9	OL
⋮	⋮	⋮	⋮	⋮

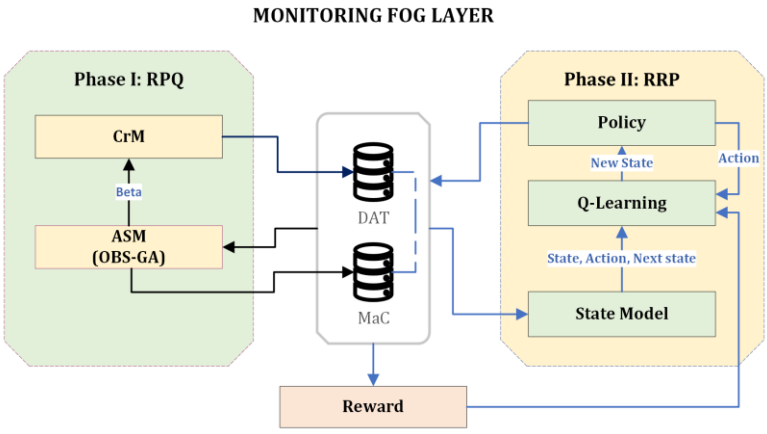


Figure 5.6: SlbmS model.

This section proposes a novel, comprehensive system for LB in fog environments. The recommendation module is based on an RL algorithm to increase the number of physical resources virtually at the ground of a fog environment. RL is an AI technique in which agents act in each environment to receive rewards. The agent gets the rewards condition of the environment and responds accordingly. The action performed causes a change in the state of the environment, which is then communicated to the agent through rewards. The RRP learns to select the optimal AP to which virtual APs are added near it and to evaluate its effects. The total RRP

phases are depicted in Algorithm 2. However, Algorithm 3 demonstrates the three fundamental stages of RL in general.

ALGORITHM 2: RL based Resource Provisioning

Output: Number of APs required to improve the quality of the cell.
Input: Overloaded cell, Set of APs, Initial values for α *and* β

1. For each new Overloaded cell do
2. Create a Q-table containing two columns {State: Available Overloaded Cell, Action: Selecting the best number of APs to be installed} initialised to zero.
3. //Taking Action
4. The agent interacts with the environment (APs) and updates the state action pairs in the Q-table Q[state, action].
5. The Q-table is used as a reference by the agent to view all possible actions (all available APs) for a given state. Then, it selects the best number of APs based on the height of LBL.
6. //Updating the Q-table
7. //Update the data in DST
8. The LBA updates the data in the DST (update the AW and the status for each FS)
9. Next

ALGORITHM 3: Q-learning based RL Algorithm

Output: Update the Q-table with the most recent Q-values.
Input: A Q-table with initial values.
Step:

1. Create a Q-table with the state and action columns initialised to zero. The agent reviews the Q-table to determine the optimal action based on the Q-value.

```
import numpy as np
# Initialize Q-table values to 0
Q = np.zeros((state_size, action_size))
```

2. //Taken Action
The agent interacts with the environment and modifies the state-action pairings contained within the Q-table Q {state, action}

```
import random# Set the percentage you wish to investigate to epsilon = 20%
if random.uniform(0, 1) < epsilon:
  """
  Explore: select a random action
  """
Else:
  """
  Exploit: select the action with max value (future rewards)
  """
```

3. // Updating the Q-table
The updates occur following each step or action and conclude when an episode is completed.

```
# Update Q-values
Q[state, action] = Q[state, action] + lr * (reward + gamma * np.max(Q[new_state, :]) — Q[state, action])
```

5.5. IMPLEMENTATION AND EVALUATION

To validate the effectiveness of the proposed algorithm, this section compares the SlbmS model and the HybOff LB algorithm using Python.

5.5.1. USED ACCESS POINTS DATASET

Contemplate Tables 5.2 and 3, cells set and their APs that will be used throughout this paper to illustrate the distinctions between the SlbmS and HybOff LB

algorithms. Table 5.3 shows the states of any cell generated under the HybOff model. We attempt to present the different cases of cells. Overloaded, balanced and underloaded cells are different cases. For that, we increased the generated IoT tasks for one cell while others received medium and low rates of generated tasks.

Table 5.3: DA-TABLE OF THE SYSTEM.

Device	$Cell$	$AT_i^{\varphi_{IoT}}$	$AT_i^{\varphi_{OP}}$	$OT_y^{\varphi_{DO}}$	OT_i^{φ}	AW_i	Status
Fn_1	3	150 Tsk/s	0 Tsk/s	70 Tsk/s	0 Tsk/s	0.2	OL
Fn_2	3	180 Tsk/s	0 Tsk/s	98 Tsk/s	0 Tsk/s	0.16	OL
Fn_3	3	110 Tsk/s	0 Tsk/s	40 Tsk/s	0 Tsk/s	0.28	OL
Fn_4	3	192 Tsk/s	0 Tsk/s	122 Tsk/s	0 Tsk/s	0.14	OL
Fn_5	3	200 Tsk/s	0 Tsk/s	120 Tsk/s	0 Tsk/s	0.15	OL
Fn_1	1	50 Tsk/s	0 Tsk/s	0 Tsk/s	1 Tsk/s	7	OL
Fn_2	1	200 Tsk/s	0 Tsk/s	50 Tsk/s	100 Tsk/s	0.12	OL
Fn_3	1	25 Tsk/s	50 Tsk/s	0 Tsk/s	1 Tsk/s	8.75	OL
Fn_1	2	4 Tsk/s	0 Tsk/s	54 $Tsk/s/s$	0 Tsk/s	3.78	UL
Fn_2	2	2 Tsk/s	0 Tsk/s	52 $Tsk/s/s$	0 Tsk/s	3.64	UL

To compute the value of MaC for each cell, the initial value for the study's variables must be initialised, such as optimised beta (β_0) and the value of TAW must first be determined. The following table shows the initialisation for our experiment:

Table 5.4: THE INITIAL VALUES OF THE PARAMETERS.

$Variables$es	β_0	Fn_i^{Cap}	AAW^{Fn}	TAW^{Fn}	MaC^{Cell}
Values	0.07	50 Tsk/s	2.42	4.08	2

According to the model's formulas, $Cell_3$ is overloaded, while $Cell_1$ is balanced, and $Cell_2$ is underloaded. Table 5.3 provides detailed information about all the APs.

5.5.2. A MATHEMATICAL RL ILLUSTRATION MODEL

Q-learning is the most prevalent reinforcement learning method, where Q represents the long-term utility of an action. Q-learning is the process of acquiring Q-values via observation. The Q-learning steps are:

Create a Q-table with two columns [State: A list of overloaded APs in the system, action: select the best AP to add extra resources nearby it] with zero-initialised values, as shown in Table 5.5. This implies that we have no information regarding the long-term rewards of each state-action combination.

Table 5.5: Q-TABLE- INITIALISED.

$State$ (s)	$Action$ (a)
0	0
0	0
0	0

As the agent learns, it performs actions denoted as 'a' in a particular state s and receives a reward r for its action. The agent also records that the system's state has

transitioned to a new state s'. The Q-value for a state-action pair $Q(s,a)$ is updated using the following formula:

$$Q(s,a) = (1 - learning_{rate})Q(s,a) + learning_{rate} \times (r + discount_{rate} \times \max_{a} Q(s',a)) \quad (5.5)$$

Here, the $learning_{rate}$ controls the weight given to new information, and the $discount_{rate}$ represents the discount factor for future rewards. The 'max' function identifies the maximum Q-value for the successive state-action pairs available to the agent.

The agent interacts with the environment (the list of overloaded APs) and modifies the state-action pairings in the Q-table $Q[state, action]$. As in Tables 2 and 3, we have ten action options ($Fn_1^1, Fn_2^1, Fn_3^1, Fn_1^2, Fn_2^2,\ Fn_1^3, Fn_2^3, Fn_3^3, Fn_4^3$, and Fn_5^3). In some states, however, action options are restricted. For instance, in state 1 (the initial state), the agent has three action options: Fn_1^1, Fn_2^1, or Fn_3^1. There are five possible actions in state three: Fn_1^3, Fn_2^3, Fn_3^3, Fn_4^3, or Fn_5^3.

When the agent takes the maximum AW, the agent receives a negative reward (-1). When it takes an average AW, it receives no reward. When it takes the lowest AW, it is rewarded with 1. Note, however, that this one-time reward differs significantly from Q-values. Indeed, we have

$$Q(Cell_1, Fn_1^1) = 0.7 * 0 + 0.2(0 + 0.8 * 0.7) = 0.11$$
$$Q(Cell_1, Fn_2^1) = 0.7 * 0 + 0.2(0 + 0.8 * 0.12) = 0.02$$
$$Q(Cell_1, Fn_3^1) = 0.7 * 0 + 0.2(0 + 0.8 * 8.75) = 1.4$$

With a learning rate of 0.2 and a discount rate of 0.8%. Fn_2 is the premium action for $Cell_1$ and Fn_4 in $Cell_5$. We notice that the algorithm chosen is loaded and overused for the cloud. Update the Q-table values according to Table 5.6.

Table 5.6: Q-TABLE-UPDATED.

State (s)	***Action*** (a)
Fn_1^1, Fn_2^1, Fn_3^1	Fn_1^1
Fn_1^2, Fn_2^2	Fn_1^2
$Fn_1^3, Fn_2^3, Fn_3^3, Fn_4^3, Fn_5^3$	Fn_5^3

5.5.3. EVALUATION METRICS

The performance of the SlbmS and HybOff LB algorithms can be compared using the metrics presented in Table 5.7. The main comparison points are Average Waiting Time ($\overline{WT}$) and the rate of offloaded tasks.

Firstly, WT is the difference in time between the Turnaround Time (TAT) and the burst time (BT), as shown in Equation 5.6. WT indicates the state of queues in the APs. To calculate the WT, TAT must be calculated first. TAT is the difference in time between the completion and arrival times. However, Average WT is the metric used to explore the HybOff model that suffers from congested cells.

$$TAT = CT - AT,$$
$$WT = CT - TAT, \tag{5.6}$$

Secondly, this counter has two kinds: external offloading (eTs) and internal offloading (iTs). The latter is used to count the processes if it targets the paired device, while external offloading is used when the target device is the cloud or other fog device in another cell. eTs reveal whether the system successfully avoids distance offloading and network congestion. The counter counts the number of offloading processes in each fog device. We prefer to keep iTs at the top level and the eTs at the bottom for each cell. Equations 5.7 and 8 depict the formulas of internal and external offloading rates:

$$iTs^{Sys} = \frac{\sum_{x=1}^{m}\sum_{y=1}^{n} OT_{xy}^{\varphi}}{\sum_{i=1}^{m} n_i}, \tag{5.7}$$

$$eTs^{Sys} = \frac{\sum_{x=1}^{m}\sum_{y=1}^{n} OT_{xy}^{\varphi_{DO}}}{\sum_{i=1}^{m} n_i}, \tag{5.8}$$

Table 5.7: EVALUATION METRICS.

Metric	Uses	Justification
$\overline{WT}$	The average execution time of the APs consumed to execute the tasks includes the queued time.	To verify the resource management efficiency of an algorithm.
iTs	It is a metric that counts, device-wise, the number of exchanged messages inside the cell.	It is a good indicator of the ability of the LB algorithm to keep the exchanged messages inside the cell.
eTs	It is a metric that counts the number of exchanged messages with external devices, including the cloud, device-wise.	It is a good indicator of the ability of the LB algorithm to keep the exchanged messages inside the cell.
$\overline{Sys^{LB}}$	It is a metric that computes the average level of LB in the system.	Using this metric allows for an excellent and fair comparison between the algorithms regarding LB.

5.5.4. SLBMS IMPLEMENTATION

Table 5.8 shows the performance of HybOff in different states. Compared to HybOff, the values are depicted in Figures 5.7 and 8, respectively.

Table 5.8: WT AND TAT FOR HYBOFF IN DIFFERENT STATES.

Model	***AWT (ms)***	***TAT (ms)***
HybOff^{B,UL}	8	4
HybOff^{OL}	3.5.5	1.55
HybOff^{SlbmS}	3	2.1

Based on Figure 5.7, it can be seen that the HybOff^{SlbmS} case has the lowest WT because the RL creates an agent that can increase the number of APs, which decreases the amount of WT significantly (Sulimani, Alghamdi et al. 2021). In addition, it has been demonstrated that the GA, used to determine the optimal value of AW, achieves the finest minimum fitness.

To verify the efficacy of HybOff^{SlbmS}, a described scenario has been analysed. In the assumed scenario, the number of resources has been changed (APs = 18), whereas the number of tasks remains constant for fair performance evaluation. Here, the HybOff algorithm is compared with and without SlbmS, representing the before and after enhancement case.

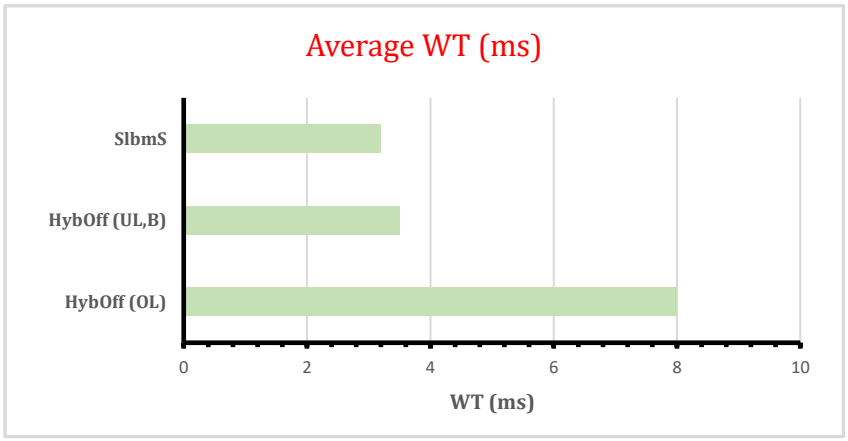


Figure 5.7: Average WT in comparison.

Table 5.8 depicts the change in the number of offloading messages, iTs and eTs, in different cases. The figure shows that external messages in HybOff (the special case) were high, and internal messages were at the bottom. Increasing the number of APs (by SlbmS recommendation) kept the congestion inside the cell. Even though the curve of the inside messages increases significantly, the congestion inside the cell protects the remaining parts of the network from unnecessary traffic.

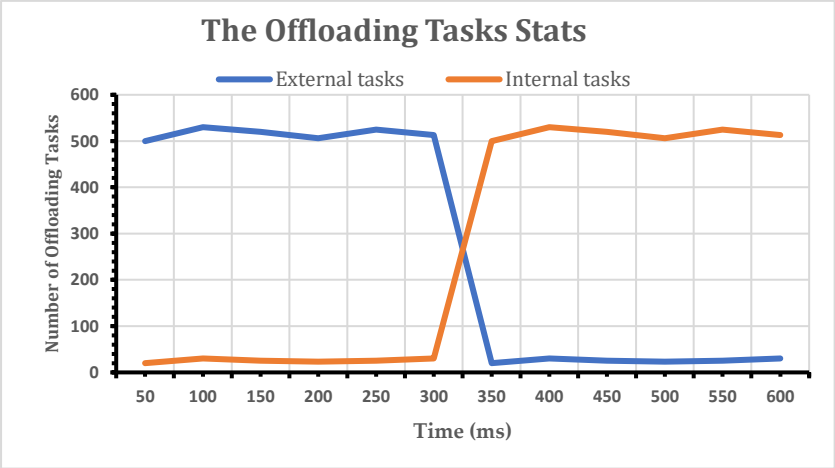


Figure 5.8: The number of exchanged tasks in the special case of HybOff and after engaging SlbmS in the solution.

5.6. Discussion

The purpose of this study was to gain a better understanding of LB in fog environments by applying the sustainability concept. The results of this chapter support the hypothesis that "Even though the software solution has a noticeable improvement for LB, hardware/software resolution will enhance the computing system in a better way." (Sulimani, Alghamdi et al. 2021). The results of this research provide supporting evidence that the system is getting better with the physical expansion of the network. Even within its case, SlbmS enables HybOff to continue giving outstanding results. There are three key findings from the current part of the research. First, reduce the average waiting time for arrival tasks by half after applying SlbmS recommendations. Second, the external offloading rate returns to normal even with the special case consequences. Third, improve the level of balancing among computing nodes.

This pattern of results is consistent with the previous literature showing that offloading is one of the optimum solutions to balance the load in FC (Aazam, Zeadally et al. 2018). Whereas past researchers have found that using artificial intelligence boosts dynamic offloading performance, this study has shown that another research dimension must be explored. These results represent the first direct demonstration of this dimension.

The comprehensive model, SlbmS, was introduced, discussed, and tested in this chapter. It consists of two systems named RPQ and RRP, where the first is responsible for improving the quality of coverage and the latter for quantity. SlbmS

has explored a new research dimension for improving LB by increasing the number of computing nodes with sufficient planning. RL has been used to provide adequate planning where the approved numbers of APs must satisfy many criteria. The new model kept network congestion and distance offloading at the bottom. The experiment uses Python, which provides a suitable simulation environment. The outcomes of this stage have shown that HybOff works efficiently with the recommendation system, SlbmS, which devastates HybOff and avoids the bottleneck of the particular case.

5.7. CONCLUSION

HybOff is introduced as a novel solution for the offloading technique used mainly to balance the load in the distributed computing environment. To empower HybOff, it is essential to fill the gap generated in some cases, such as overload cells. Leaving HybOff in this state, exposure to those high-probability situations causes an apparent decrease in efficiency. However, this chapter presents the second and last phase of this dissertation. SlbmS is the comprehensive solution for HybOff that introduces a competitive solution for the problem of time-sensitive applications in large-scale fog networks. The idea behind SlbmS is how to increase the physical quality and quantity of computing nodes with sufficient planning. It works through two sequential phases, each concerned with one side compatible with the sustainable concept. The proposed solution utilises the genetic algorithm in phase one, while RL is in phase two. The experimental outcomes by Python show an apparent enhancement in the system performance. The experimental result shows a return of HybOff to its original performance. This work is the first to engage the precise meaning of sustainable concepts within LB in Fog environments. Although this work has some limitations, future work is essential to find a better way to reintroduce this work.

6. CHAPTER, CONCLUSION AND FUTURE RESEARCH

***Abstract*:** In this thesis, we have planned to present five articles that have contributed to the development of LB in FC, including improving service placement policy, a novel offloading algorithm, and a comprehensive solution that seeks to support a long-term solution in FC. In Chapter One, we depict the thesis framework and the elements for this research, while we analyse the extant FC literature from theoretical and practical perspectives in Chapter Two. Then, we presented three of our most representative works in FC in Chapters Three to Five, respectively. **However,** this chapter presents the research phases, proposed solutions, and their limitations on the words. Moreover, future works are offered, including the technical contributions of this thesis.

6.1. INTRODUCTION

This study sought to identify effective LB strategies in a fog environment. Based on experimental analysis of the fog model under different conditions, it can be concluded that a hybrid offloading technique with a sustainability concept is crucial when designing a distributed computing system. This proposed design aims to override the dilemma of LB in a fog environment. The outcomes indicate that fog devices are running balanced with minimal network traffic.

The primary research questions for this study were as follows:

- **Research Question 1:** What is the required technique to minimise the consumed time by offloading the decision process?
- **Research Question 2:** What is the required strategy to minimise the number of offloading messages?
- **Research Question 3:** What is the suitable strategy to avoid the distant offloading phenomena?
- **Research Question 4:** What is a new solution for optimal placement policy over fog physical resources to overcome the inherent issues of prevalent resource allocation mentioned in RQ1, RQ2, and RQ3?
- **Research Question 5:** What is a new solution for optimal offloading over fog physical resources to overcome the inherent issues of prevalent resource allocation mentioned in RQ1, RQ2, and RQ3?
- **Research Question 6:** How could the performance of the proposed systems be evaluated in a fog environment in RQ4 and RQ5?
- **Research Question 7:** What are the limitations of the optimum LB solution in this research?
- **Research Question 8:** How should reinforcement learning be designed to apply sufficient planning before expansion?
- **Research Question 9:** How to design a comprehensive model to monitor the entire system and suggest a recommendation to tackle the overload situation by proposing hardware modifications?

6.1.1. RESEARCH OVERVIEW

In the earlier times of this research, the research gap and the general framework were proposed; it consists of two phases. The first phase enhances the LB level, while the second phase provides the sustainability concept. As a result, the general framework was identified and approved to be constructed in two phases.

However, there are many approaches to solving the problem of LB in FC. We explore the optimum solution between offloading and service placement

approaches to enhance the research problem. This process is the output of the first phase. In the following lines, we present the methodology used to solve these questions:

- **The methodology applied to answer question 1:** The offloading decision is the process required to find the optimum accessible server to offload the exceeded tasks. It consumes a lot of time. All prevalent offloading follows the same behaviour, while static offloading does not require time to decide. However, this research has used hybrid offloading to eliminate this problem. HybOff is the actual implementation of hybrid offloading. The algorithm depends mainly on the clustering technique to give the system a limited number of fog servers.
- **The methodology applied to answer question 2:** The offloading decision generates an uncontrolled number of exchanged messages. We utilise two techniques to reduce the number of exchanged offloading messages. Firstly, hybrid behaviour, which operates static behaviour, uses a limited number of messages to update the servers' information. Moreover, the clustering technique can limit the messages inside the generated cell.
- **The methodology applied to answer question 3:** The nature of the network and the offloading algorithm make the offloading to a distant server possible. To overcome this problem, HybOff utilises the cell structure to hold the messages inside the cell.
- **The methodology applied to answer question 4:** Placement policy is one of the techniques used to improve the level of LB. In this technique, we consider the number of servers in the shortest path to the cloud as one cell. Thus, RODSPP uses the shortest path toward the cloud instead of the clustering technique. The father's server forwards to its child. However, RODSPP outperforms its peers and dramatically improves LB.
- **The methodology applied to answer question 5:** This part attempts to respond to the fifth research question, "What is a new solution for optimal offloading over fog physical resources to overcome the inherent issues of prevalent offloading mentioned in RQ1, RQ2, and RQ3?". It relies heavily on

clustering theory and hybrid offloading. This technique's primary contributions are reviving static offloading and improving the efficacy of IoT applications, particularly time-sensitive ones, regardless of network size.

- **The methodology applied to answer question 6:** Measuring the performance of the proposed algorithms is the only way to judge the optimum solution. Utilisation and LBLs are critical metrics that aim to evaluate them.
- **The methodology applied to answer question 7:** The main drawback of HybOff is that if a cell encounters many arrival tasks and is entirely loaded, HybOff achieves unanticipated results; in this case, Hybrid offloading provides no benefit. All computing nodes may reach the utilisation limit, allowing the disabOff() function to block the offloading feature inside the cell. Consequentiality, HybOff, in this case, will start the offloading process across cells or the cloud to find free resources that might be located a distance away; in this scenario, HybOff will follow prevalent offloading, which, as mentioned, has drawbacks. It generates undesirable phenomena, such as network congestion and distant offloading. Accordingly, HybOff cannot work efficiently within overloaded cells.
- **The methodology applied to answer question 8:** Increasing the ground resources is the only way to overcome the limitations of HybOff. Sufficient planning for network expansion is a suitable approach to avoid network oversizing and congestion. Reinforcement learning works to study the effect of increasing the number of fog servers virtually or offline. By analysing several arrivals and offloaded tasks, we can identify the unused servers to move them to the loaded area.
- **The methodology applied to answer question 9:** By answering this question, we present a SlbmS that seeks to find a comprehensive solution for a HybOff model that has suffered from the increased demand for data processing in the highly loaded cells and, on the other hand, their limited physical resources. SlbmS is designed to satisfy the requirements of time-sensitive applications using machine learning to acquire the sustainable concept. According to the framework, SlbmS constructs in two sequential stages to achieve the research

aim. The investigation examines the performance of HybOff with and without SlbmS in congested cells.

6.1.2. Research Phases

The *first phase* consists of three stages. In stage one, chapter four, we focused on studying the service placement technique, which enhances the LB in the fog layer. Consequently, reinforcement optimisation for a decentralised service placement policy (RODSPP) was proposed for fog networks that leverage service placement theory. Later and in chapter five, a novel hybrid offloading (HybOff) algorithm was proposed to overcome the inherent issues of prevalent offloading algorithms, representing stage two in the research.

The *third stage in phase one* seeks to adopt the ideal technique to balance the load with the minimum cost. Both the proposed algorithms enhanced the level of the research problem. Even though the service placement technique has an outstanding outcome compared to its peers, it costs the network an immense number of exchanged messages due to its attitude, which relies substantially on offloading, considered a hidden cost, to find the required service. RODSPP cannot be controlled to reduce the traffic in the network. On the other hand, HybOff seems to be an ideal solution due to its solvable disadvantages that do not conflict with its essential attitude. Hence, HybOff is the approved pilot solution for the second phase.

However, the experimental results of phase one reveal that HybOff follows the attitude of prevalent solutions in some unavoidable cases. However, the second and last phase of the thesis research aims to create a comprehensive solution for a HybOff model that has suffered from the increased demand for data processing in the highly loaded cells and, on the other hand, the limited physical resources to satisfy the requirements of time-sensitive applications. Thus, this work proposes a SlbmS that seeks a long-term concept using machine learning.

According to the framework, SlbmS constructs in two sequential stages to achieve the research aim. While enhancing the coverage quality of the fog layer is the outcome of the first stage, proposing recommendations to strengthen the resource provisioning of the fog layer is the result of the second stage. The explicit goal of SlbmS is to decrease the number of offloading processes in the fog layer by

improving the coverage quality and quantity of fog devices, which can be implicitly considered resource provisioning. Squeezing offloading procedures into the system enables it to override the network-congested phenomenon.

To verify the efficiency of the proposed model, a proper simulation experiment was held using MATLAB. The investigation examines the performance of HybOff with and without SlbmS in congested cells. The experiment shows that when the proposed recommendation of quality or quantity was implemented, HybOff enormously benefited. Sufficient planning for the expansion process allows HybOff to overcome the problems of exceptional cases. The results reveal that HybOff reverted to its original performance.

The remainder of this chapter is structured as follows: Section two concludes the techniques used in this research. The research limitations are presented in Section Three. Section four discusses possible future works.

6.2. Load Balancing Solutions

In Chapter 4, we examined the problem of LB in fog networks using service placement policy; RODSPP is a reinforcement optimisation for a decentralised service placement policy (RODSPP) scheme for fog networks that leverages service placement theory. It represents stage one (in phase one) to examine the performance of service placement police to balance the load. By matching the overhead of the fog nodes to the cloud's shortest route, the modified placement policy outperforms its peers significantly. RODSPP follows the vertical dimension toward the cloud in its solution. RODSPP reduces latency and computing consumption by 24.1% and 4%, respectively, in task performance, as demonstrated by simulation and trace-driven evaluation outcomes. In contrast, its computational complexity has decreased significantly over time.

Chapter Five examines how the offloading theory can assist fog nodes in burden sharing. We have proposed the offloading in the horizontal dimension, upon which our hypothesis relies, no vertical offloading except to the cloud centre. In this stage, HybOff is the proposed offloading algorithm. The hybrid offloading algorithm aims to increase the periphery computing utilisation rate. It executes sensitive tasks locally while sending other tasks to neighbouring nodes. Clustering techniques

reduce communication bandwidth costs by preventing outsourcing to remote devices. On the other hand, hybrid offloading combines the benefits of static and dynamic offloading, thereby reducing the time required to identify the optimal node for offloading. The outcomes are exceptional compared to the service placement policy implemented through vertical offloading.

Thereby, stage three selects HybOff to represent phase one, whose primary objective is to ensure that all fog nodes in their layer perform the same quantity of tasks. Extensive stage two experiments demonstrate that HybOff confronts significant challenges in continuing to show outstanding results. The challenges manifest when most fog nodes are loaded in specific cells. The laden nodes will transfer their weighty tasks to other cells or the cloud at that moment. However, this behaviour attempts to replicate the traditional offloading pattern. Therefore, the prototype HybOff algorithm requires further reinforcement from another system. Fortunately, stage two relies on the concept of sustainability. Accordingly, stage two is modified to address the HybOff issue.

In Chapter Six, we discussed SlbmS, a LB monitoring system that enables fog networks to share workload and offload computations equitably. The SlbmS incorporates the hybrid offloading theory and the sustainable notion to create a robust LB model for a fog environment. We have devised the system model of computation offloading and formulated the LB problem to maximise fairness. Adopting the 'two-stage' paradigm, we have presented a comprehensive algorithm design for LB and conducted exhaustive evaluation studies by performing simulations and experiments on various cases. The experimental findings indicate that SlbmS effectively achieves LB with a guaranteed performance of a near-optimal fairness index of up to 0.85 and a 50 % improvement over conventional baseline methods.

6.3. Model Limitations in the real world

This research seeks to create a competitive solution for LB issues in a fog environment. However, as a requirement for academic research, we must mention the limitations noticed during our study to pave the way for the following

researchers. The following points depict the most significant restrictions and drawbacks of this work:

1. **Cost:** In the real world, using this solution requires a physical modification in the position of APs to benefit from this work. Generally, the enterprise performs the network expansion through its team or a contractor. This action costs the enterprise human resources to reinstall or install the computing nodes. Despite the cost, the development of the network remains the ultimate solution for the research topic.
2. **Distrust:** In the real world, many enterprises have a lower level with the automated recommendations systems to be applied. They trust the second party, the consultant, to decide the expansion process**.**
3. **Interruption of services:** Reinstalling or installing some APs might cause a denial of service. Reinstallation of some APs will disconnect some clients from the system. The enterprise must reposition APs with many users near zero regarding arrival and offloading tasks. Fortunately, the expansion process occurs typically rarely.
4. **Installation time:** It is the time required to install, and, in some rare cases, the installation process may take longer, which may extend the service interruption time.

Despite these limitations, this research can be recognised as a first step toward integrating the study aims, LB in FC and sustainability, which have yet to be directly linked to our knowledge.

Moreover, this study has enhanced our understanding of the rapport between LB in FC and sustainability. We hope that the current research will stimulate further investigation of this vital area.

Therefore, it contributes to an embarking body of evidence suggesting that sufficient planning in the expanding process is required. Although the generality of the current results must be established by future research, this study has supported the software/hardware solution. Not only was the software solution the key to LB's dilemma in a fog environment.

6.4. FUTURE WORKS

The following points highlight some of the future research topics that are considered to be hot spots of current research:

Though the proposed algorithm (RODSPP) achieves the best performance within the requirements considered in this study, we plan to improve upon these results by adding more dimensions to the service placement policy. Thus, we plan to work toward a near-zero-delay system by combining static and dynamic offloading mechanisms for LB in FC.

In the future, we will investigate the optimal number of cell servers to reduce energy consumption. Since this work concentrates only on the outsourcing decision-making process, we intend to examine the issue of task migration and virtual machine migration in the future. Thus, after the initial decision for the task has been made, the task migration will be determined based on the circumstances.

In addition, iFogSim, a Java-based framework for modelling and simulating FC infrastructures and services, will be utilised to simulate relevant experiments. We intend to affect the task migration and mobility issue using the Markov decision process, the deep learning method, and actual datasets.

We intend to analyse the negative and positive effects of VM migrations in future work. In addition, we will create a method to balance the negative and positive impacts of service migrations.

Location-awareness: future research needs to study to include the nearby nodes by expanding the coverage of cells. This action required us to be aware of the physical location of all APs.

BIBLIOGRAPHY

Aazam, M., S. Zeadally and K. A. Harras (2018). "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities." Future Generation Computer Systems **87**: 278-289.

Abedin, S. F., A. K. Bairagi, M. S. Munir, N. H. Tran and C. S. Hong (2018). "Fog load balancing for massive machine type communications: A game and transport theoretic approach." IEEE Access **7**: 4204-4218.

Al-Tarawneh, M. A. (2021). "Bi-objective optimization of application placement in fog computing environments." Journal of Ambient Intelligence and Humanized Computing: 1-24.

Albalawi, M., E. Alkayal, A. Barnawi and M. Boulares (2022). "Load Balancing Based on Many-objective Particle Swarm Optimization Algorithm with Support Vector Regression in Fog Computing." Journal of Engineering and Applied Sciences Technology. SRC/JEAST-170. DOI: doi. org/10.47363/JEAST/2022 (4) **138**.

Alsharif, M. H., A. Jahid, A. H. Kelechi and R. Kannadasan (2023). "Green IoT: A review and future research directions." Symmetry **15**(3): 757.

Apat, H. K., R. Nayak and B. Sahoo (2023). "A comprehensive review on Internet of Things application placement in Fog computing environment." Internet of Things: 100866.

Arora, U. and N. Singh (2021). "IoT application modules placement in heterogeneous fog–cloud infrastructure." International Journal of Information Technology: 1-8.

Arshad, H., H. A. Khattak, M. A. Shah, A. Assad and Z. Ameer (2018). "Evaluation and analysis of bio-inspired optimization techniques for bill estimation in fog computing." International Journal of Advanced Computer Science and Applications **9**(7).

Baburao, D., T. Pavankumar and C. Prabhu (2021). "Load balancing in the fog nodes using particle swarm optimization-based enhanced dynamic resource allocation method." Applied Nanoscience: 1-10.

Baranwal, G. and D. P. Vidyarthi (2021). "FONS: a fog orchestrator node selection model to improve application placement in fog computing." The Journal of Supercomputing **77**: 10562-10589.

Beraldi, R. and H. Alnuweiri (2019). Exploiting power-of-choices for load balancing in fog computing. 2019 IEEE International Conference on Fog Computing (ICFC), IEEE.

Besharati, R., M. H. Rezvani and M. M. Gilanian Sadeghi (2023). "An Auction-Based Bid Prediction Mechanism for Fog-Cloud Offloading Using Q-Learning." Complexity **2023**.

Bisht, J. and V. Subrahmanyam "Survey on Load Balancing and Scheduling Algorithms in Cloud Integrated Fog Environment."

Bittencourt, L. F., J. Diaz-Montes, R. Buyya, O. F. Rana and M. Parashar (2017). "Mobility-aware application scheduling in fog computing." IEEE Cloud Computing **4**(2): 26-35.

Bolarinwa, O. A. (2015). "Principles and methods of validity and reliability testing of questionnaires used in social and health science researches." Nigerian Postgraduate Medical Journal **22**(4): 195.

Bonomi, F., R. Milito, J. Zhu and S. Addepalli (2012). Fog computing and its role in the internet of things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing.

Brereton, P., B. A. Kitchenham, D. Budgen, M. Turner and M. Khalil (2007). "Lessons from applying the systematic literature review process within the software engineering domain." Journal of systems and software **80**(4): 571-583.
Brogi, A., S. Forti, C. Guerrero and I. Lera (2020). "How to place your apps in the fog: State of the art and open challenges." Software: Practice and Experience **50**(5): 719-740.
Burhan, M., H. Alam, A. Arsalan, R. A. Rehman, M. Anwar, M. Faheem and M. W. Ashraf (2023). "A Comprehensive Survey on the Cooperation of Fog Computing Paradigm-based IoT Applications: Layered Architecture, Real-Time Security Issues, and Solutions." IEEE Access.
Buyya, R., S. Pandey and C. Vecchiola (2009). Cloudbus toolkit for market-oriented cloud computing. IEEE International Conference on Cloud Computing, Springer.
Cao, B., M. Li, X. Liu, J. Zhao, W. Cao and Z. Lv (2021). "Many-objective deployment optimization for a drone-assisted camera network." IEEE transactions on network science and engineering **8**(4): 2756-2764.
Cao, B., Z. Sun, J. Zhang and Y. Gu (2021). "Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing." IEEE Transactions on Intelligent Transportation Systems **22**(6): 3832-3840.
Cao, B., J. Zhang, X. Liu, Z. Sun, W. Cao, R. M. Nowak and Z. Lv (2021). "Edge–cloud resource scheduling in space–air–ground-integrated networks for internet of vehicles." IEEE Internet of Things Journal **9**(8): 5765-5772.
Cao, B., J. Zhao, Y. Gu, S. Fan and P. Yang (2019). "Security-aware industrial wireless sensor network deployment optimization." IEEE transactions on industrial informatics **16**(8): 5309-5316.
Cao, B., J. Zhao, Y. Gu, Y. Ling and X. Ma (2020). "Applying graph-based differential grouping for multiobjective large-scale optimization." Swarm and Evolutionary Computation **53**: 100626.
Cao, B., J. Zhao, P. Yang, Y. Gu, K. Muhammad, J. J. Rodrigues and V. H. C. de Albuquerque (2019). "Multiobjective 3-D topology optimization of next-generation wireless data center network." IEEE Transactions on Industrial Informatics **16**(5): 3597-3605.
Cao, K., B. Wang, H. Ding, L. Lv, J. Tian, H. Hu and F. Gong (2021). "Achieving reliable and secure communications in wireless-powered NOMA systems." IEEE Transactions on Vehicular Technology **70**(2): 1978-1983.
Catruc, I. and D. Iosifescu (2020). "IoT Integration with Mobile and Cloud Solutions." Informatica Economica **24**(2).
Celesti, A., M. Fazio, M. Villari and A. Puliafito (2012). "Virtual machine provisioning through satellite communications in federated Cloud environments." Future Generation Computer Systems **28**(1): 85-93.
Chakraborty, S. and K. Mazumdar (2023). "A Hybrid GRASP-GA based collaborative task offloading technique in fog computing." Multimedia Tools and Applications: 1-30.
Chandak, A. and N. K. Ray (2019). A review of load balancing in fog computing. 2019 International Conference on Information Technology (ICIT), IEEE.
Chen, C.-M., Y. Huang, K.-H. Wang, S. Kumari and M.-E. Wu (2021). "A secure authenticated and key exchange scheme for fog computing." Enterprise Information Systems **15**(9): 1200-1215.
Cheng, B., M. Wang, S. Zhao, Z. Zhai, D. Zhu and J. Chen (2017). "Situation-aware dynamic service coordination in an IoT environment." IEEE/ACM Transactions On Networking **25**(4): 2082-2095.

Colistra, G., V. Pilloni and L. Atzori (2014). "The problem of task allocation in the Internet of Things and the consensus-based approach." Computer Networks **73**: 98-111.
Dai, X., Z. Xiao, H. Jiang, M. Alazab, J. C. Lui, S. Dustdar and J. Liu (2022). "Task co-offloading for d2d-assisted mobile edge computing in industrial internet of things." IEEE Transactions on Industrial Informatics **19**(1): 480-490.
Danielsson, P.-E. (1980). "Euclidean distance mapping." Computer Graphics and image processing **14**(3): 227-248.
Das, R. and M. M. Inuwa (2023). "A review on fog computing: issues, characteristics, challenges, and potential applications." Telematics and Informatics Reports: 100049.
Datta, S. K. and C. Bonnet (2017). An edge computing architecture integrating virtual IoT devices. 2017 IEEE 6th Global Conference on Consumer Electronics (GCCE), IEEE.
Deepak, G. and B. Pradeep (2012). "Challenging issues and limitations of mobile computing." International Journal of Computer Technology & Applications **3**(1): 177-181.
Dhyani, D. (2023). E-Health data risks & protection for public cloud: An elderly healthcare usecase for Swedish municipality.
Ebrahim, M. and A. Hafid (2023). "Privacy-Aware Load Balancing in Fog Networks: A Reinforcement Learning Approach." arXiv preprint arXiv:2301.09497.
Ferrer, A. J., J. M. Marquès and J. Jorba (2019). "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing." ACM Computing Surveys (CSUR) **51**(6): 1-36.
Gallupe, R. B. (2007). "The tyranny of methodologies in information systems research1." ACM SIGMIS Database: the DATABASE for Advances in Information Systems **38**(3): 20-28.
Ghanbari, H., B. Simmons, M. Litoiu and G. Iszlai (2012). "Feedback-based optimization of a private cloud." Future Generation Computer Systems **28**(1): 104-111.
Goel, A., L. Abeni, C. Krasic, J. Snow and J. Walpole (2002). "Supporting time-sensitive applications on a commodity OS." ACM SIGOPS Operating Systems Review **36**(SI): 165-180.
Goel, G. and A. K. Chaturvedi (2023). A Systematic Review of Task Offloading & Load Balancing Methods in a Fog Computing Environment: Major Highlights & Research Areas. 2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT), IEEE.
Gomes, E., F. Costa, C. De Rolt, P. Plentz and M. Dantas (2021). A survey from real-time to near real-time applications in fog computing environments. Telecom, MDPI.
Gonzalez-Mejía, A. M., T. N. Eason, H. Cabezas and M. T. Suidan (2012). "Assessing sustainability in real urban systems: the greater Cincinnati metropolitan area in Ohio, Kentucky, and Indiana." Environmental science & technology **46**(17): 9620-9629.
González-Mejía, A. M., T. N. Eason, H. Cabezas and M. T. Suidan (2014). "Social and economic sustainability of urban systems: comparative analysis of metropolitan statistical areas in Ohio, USA." Sustainability science **9**: 217-228.
Gowri, V. and B. Baranidharan (2023). "Multi Objective Hybrid Load Balancing Based Optimization Algorithm for Improving Fog Computing Performance."
Guerrero (2019). "A lightweight decentralized service placement policy for performance optimization in fog computing. J. Ambient Intell. Humanized Comput. 10 (6), 2435–2452 (2019)."
Guerrero, C., I. Lera and C. Juiz (2019). "A lightweight decentralized service placement policy for performance optimization in fog computing. J. Ambient Intell. Humanized Comput. 10 (6), 2435–2452 (2019)."

Guerrero, C., I. Lera and C. Juiz (2019). A lightweight decentralized service placement policy for performance optimization in fog computing. J. Ambient Intell. Humanized Comput. 10 (6), 2435–2452 (2019).
Guo, F., W. Zhou, Q. Lu and C. Zhang (2022). "Path extension similarity link prediction method based on matrix algebra in directed networks." Computer Communications **187**: 83-92.
Gupta, A. and S. K. Gupta (2022). "A survey on green unmanned aerial vehicles-based fog computing: Challenges and future perspective." Transactions on Emerging Telecommunications Technologies **33**(11): e4603.
Gupta, A. and S. K. Gupta (2022). "A survey on green unmanned aerial vehicles-based fog computing: Challenges and future perspective." Transactions on Emerging Telecommunications Technologies: e4603.
Gupta, H., A. Vahid Dastjerdi, S. K. Ghosh and R. Buyya (2017). "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments." Software: Practice and Experience **47**(9): 1275-1296.
Haris, M. and R. Z. Khan (2018). "A systematic review on cloud computing." International Journal of Computer Sciences and Engineering **6**(11): 632-639.
Hassan, H. O., S. Azizi and M. Shojafar (2020). "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments." IET communications **14**(13): 2117-2129.
He, X., H. Lu, H. Huang, Y. Mao, K. Wang and S. Guo (2020). "QoE-based cooperative task offloading with deep reinforcement learning in mobile edge networks." IEEE Wireless Communications **27**(3): 111-117.
Hermann, S. D., M. Emmelmann, O. Belaifa and A. Wolisz (2007). Investigation of IEEE 802.11 k-based access point coverage area and neighbor discovery. 32nd IEEE Conference on Local Computer Networks (LCN 2007), IEEE.
Hou, X., Y. Li, M. Chen, D. Wu, D. Jin and S. Chen (2016). "Vehicular fog computing: A viewpoint of vehicles as the infrastructures." IEEE Transactions on Vehicular Technology **65**(6): 3860-3873.
Huang, Z., K.-J. Lin, S.-Y. Yu and J. Y.-j. Hsu (2014). "Co-locating services in IoT systems to minimize the communication energy cost." Journal of Innovation in Digital Ecosystems **1**(1-2): 47-57.
Hussein, M. K. and M. H. Mousa (2020). "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization." IEEE Access **8**: 37191-37201.
Iorga, M., L. Feldman, R. Barton, M. Martin, N. Goren and C. Mahmoudi (2017). The nist definition of fog computing, National Institute of Standards and Technology.
Jamil, B., M. Shojafar, I. Ahmed, A. Ullah, K. Munir and H. Ijaz (2020). "A job scheduling algorithm for delay and performance optimization in fog computing." Concurrency and Computation: Practice and Experience **32**(7): e5581.
Janjic, V. (2012). Load balancing of irregular parallel applications on heterogeneous computing environments, University of St Andrews.
Javaid, N., A. A. Butt, K. Latif and A. Rehman (2019). Cloud and fog based integrated environment for load balancing using cuckoo levy distribution and flower pollination for smart homes. 2019 International Conference on Computer and Information Sciences (ICCIS), IEEE.
Jaya, I. (2023). "Resource allocation in cloud gaming."

Jebur, T. K. (2023). "Greening the internet of things: A comprehensive review of sustainable iot solutions from an educational perspective." Indonesian Journal of Educational Research and Technology **3**(3): 247-256.
Jiang, H., X. Dai, Z. Xiao and A. K. Iyengar (2022). "Joint task offloading and resource allocation for energy-constrained mobile edge computing." IEEE Transactions on Mobile Computing.
Jiang, H., Z. Xiao, Z. Li, J. Xu, F. Zeng and D. Wang (2020). "An energy-efficient framework for internet of things underlaying heterogeneous small cell networks." IEEE Transactions on Mobile Computing **21**(1): 31-43.
Jiang, Y.-L., Y.-S. Chen, S.-W. Yang and C.-H. Wu (2018). "Energy-efficient task offloading for time-sensitive applications in fog computing." IEEE Systems Journal **13**(3): 2930-2941.
Jiang, Y., C. Li, Y. Zhang, R. Zhao, K. Yan and W. Wang (2021). "Data-driven method based on deep learning algorithm for detecting fat, oil, and grease (FOG) of sewer networks in urban commercial areas." Water Research **207**: 117797.
Kadhim, A. J. and S. A. H. Seno (2018). "Maximizing the utilization of fog computing in internet of vehicle using SDN." IEEE Communications Letters **23**(1): 140-143.
Kaelbling, L. P., M. L. Littman and A. W. Moore (1996). "Reinforcement learning: A survey." Journal of artificial intelligence research **4**: 237-285.
Kaiwartya, O., A. H. Abdullah, Y. Cao, J. Lloret, S. Kumar, R. R. Shah, M. Prasad and S. Prakash (2017). "Virtualization in wireless sensor networks: Fault tolerant embedding for internet of things." IEEE Internet of Things Journal **5**(2): 571-580.
Kamel, M. A., X. Yu and Y. Zhang (2020). "Formation control and coordination of multiple unmanned ground vehicles in normal and faulty situations: A review." Annual reviews in control **49**: 128-144.
Kashani, M. H., A. Ahmadzadeh and E. Mahdipour (2020). "Load balancing mechanisms in fog computing: A systematic review." arXiv preprint arXiv:2011.14706.
Kashani, M. H. and E. Mahdipour (2022). "Load balancing algorithms in fog computing: A systematic review." IEEE Transactions on Services Computing.
Kaur, K. and M. Sachdeva (2020). "Fog computing in IoT: An overview of new opportunities." Proceedings of ICETIT 2019: 59-68.
Kaur, M. and R. Aron (2021). "A systematic study of load balancing approaches in the fog computing environment." The Journal of supercomputing **77**(8): 9202-9247.
Khan, A. A., A. A. Laghari, Z. A. Shaikh, Z. Dacko-Pikiewicz and S. Kot (2022). "Internet of Things (IoT) security with blockchain technology: A state-of-the-art review." IEEE Access.
Khan, K. S., G. Ter Riet, J. Glanville, A. J. Sowden and J. Kleijnen (2001). Undertaking systematic reviews of research on effectiveness: CRD's guidance for carrying out or commissioning reviews, NHS Centre for Reviews and Dissemination.
Khosroabadi (2021). "SCATTER: Service Placement in Real-Time Fog-Assisted IoT Networks." Journal of Sensor and Actuator Networks **10**(2): 26.
Khosroabadi, F., F. Fotouhi-Ghazvini and H. Fotouhi (2021). "SCATTER: Service Placement in Real-Time Fog-Assisted IoT Networks." Journal of Sensor and Actuator Networks **10**(2): 26.
Kitchenham, B. (2004). "Procedures for performing systematic reviews." Keele, UK, Keele University **33**(2004): 1-26.
Klas, G. I. (2015). "Fog computing and mobile edge cloud gain momentum open fog consortium, etsi mec and cloudlets." Google Scholar **1**(1): 1-13.

Kriushanth, M., L. Arockiam and G. J. Mirobi (2013). "Auto scaling in Cloud Computing: an overview." International Journal of Advanced Research in Computer and Communication Engineering **2**(7): 2278-1021.
Kuempel, C. D., V. M. Adams, H. P. Possingham and M. Bode (2018). "Bigger or better: the relative benefits of protected area network expansion and enforcement for the conservation of an exploited species." Conservation Letters **11**(3): e12433.
Kumar, M. G. V., S. Karunakaran, S. Chandre, R. K. Godi, P. Manirajkumar and A. Balaram (2023). "Implementation of Microgrid Digital Twin System for Unmanned Vehicles with Cloud Computing Techniques." SN Computer Science **4**(5): 566.
Li, C., H. Zhuang, Q. Wang and X. Zhou (2018). "SSLB: self-similarity-based load balancing for large-scale fog computing." Arabian Journal for Science and Engineering **43**(12): 7487-7498.
Likas, A., N. Vlassis and J. J. Verbeek (2003). "The global k-means clustering algorithm." Pattern recognition **36**(2): 451-461.
Lin, B.-S., B. Kar, T.-L. Chin, Y.-D. Lin and C.-Y. Chen (2023). "Cost optimization of cloud-edge-fog federated systems with bidirectional offloading: one-hop versus two-hop." Telecommunication Systems: 1-19.
Liu, W., Y. Xu, N. Qi, K. Yao, Y. Zhang and W. He (2020). Joint computation offloading and resource allocation in UAV swarms with multi-access edge computing. 2020 International Conference on Wireless Communications and Signal Processing (WCSP), IEEE.
Lu, C., J. Zheng, L. Yin and R. Wang (2023). "An improved iterated greedy algorithm for the distributed hybrid flowshop scheduling problem." Engineering Optimization: 1-19.
Lu, H., C. Gu, F. Luo, W. Ding and X. Liu (2020). "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning." Future Generation Computer Systems **102**: 847-861.
Lu, S., J. Wu, N. Wang, Y. Duan, H. Liu, J. Zhang and J. Fang (2023). "Resource provisioning in collaborative fog computing for multiple delay-sensitive users." Software: Practice and Experience **53**(2): 243-262.
Lv, Z., J. Wu, Y. Li and H. Song (2022). "Cross-layer optimization for industrial Internet of Things in real scene digital twins." IEEE Internet of Things Journal **9**(17): 15618-15629.
Mach, P. and Z. Becvar (2017). "Mobile edge computing: A survey on architecture and computation offloading." IEEE communications surveys & tutorials **19**(3): 1628-1656.
Madakam, S., V. Lake, V. Lake and V. Lake (2015). "Internet of Things (IoT): A literature review." Journal of Computer and Communications **3**(05): 164.
Mahmud, R., R. Kotagiri and R. Buyya (2018). "Fog computing: A taxonomy, survey and future directions." Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives: 103-130.
Maiti, P., B. Sahoo, A. K. Turuk, A. Kumar and B. J. Choi (2021). "Internet of Things applications placement to minimize latency in multi-tier fog computing framework." ICT Express.
Mao, Y., C. You, J. Zhang, K. Huang and K. B. Letaief (2017). "A survey on mobile edge computing: The communication perspective." IEEE communications surveys & tutorials **19**(4): 2322-2358.
Martinez, M. N. and M. J. Bartholomew (2017). "What does it "mean"? A review of interpreting and calculating different types of means and standard deviations." Pharmaceutics **9**(2): 14.

Maswood, M. M. S., M. R. Rahman, A. G. Alharbi and D. Medhi (2020). "A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment." IEEE Access **8**: 113737-113750.
Meurisch, C., A. Seeliger, B. Schmidt, I. Schweizer, F. Kaup and M. Mühlhäuser (2015). Upgrading wireless home routers for enabling large-scale deployment of cloudlets. Mobile Computing, Applications, and Services: 7th International Conference, MobiCASE 2015, Berlin, Germany, November 12–13, 2015, Revised Selected Papers 7, Springer.
Mon, M. M. and M. A. Khine (2019). Scheduling and load balancing in cloud-fog computing using swarm optimization techniques: A survey, MERAL Portal.
Morkevicius, N., A. Venčkauskas, N. Šatkauskas and J. Toldinas (2021). "Method for Dynamic Service Orchestration in Fog Computing." Electronics **10**(15): 1796.
Mouradian, C., D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow and P. A. Polakos (2017). "A comprehensive survey on fog computing: State-of-the-art and research challenges." IEEE communications surveys & tutorials **20**(1): 416-464.
Mukherjee, M., L. Shu and D. Wang (2018). "Survey of fog computing: Fundamental, network applications, and research challenges." IEEE Communications Surveys & Tutorials **20**(3): 1826-1857.
Muniswamaiah, M., T. Agerwala and C. C. Tappert (2021). A Survey on Cloudlets, Mobile Edge, and Fog Computing. 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.
Mutlag, A. A., M. K. Abd Ghani, O. Mohd, K. H. Abdulkareem, M. A. Mohammed, M. Alharbi and Z. J. Al-Araji (2023). "A new fog computing resource management (FRM) model based on hybrid load balancing and scheduling for critical healthcare applications." Physical Communication **59**: 102109.
Naha, R. K., S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang and R. Ranjan (2018). "Fog computing: Survey of trends, architectures, requirements, and research directions." IEEE access **6**: 47980-48009.
Neto, E. C. P., G. Callou and F. Aires (2017). An algorithm to optimise the load distribution of fog environments. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE.
Ningning, S., G. Chao, A. Xingshuo and Z. Qiang (2016). "Fog computing dynamic load balancing mechanism based on graph repartitioning." China Communications **13**(3): 156-164.
Niu, L., J. Lu and G. Zhang (2009). "Cognition-driven decision support for business intelligence." Models, Techniques, Systems and Applications. Studies in Computational Intelligence, Springer, Berlin: 4-5.
Nokia, I. (2013). "Increasing mobile operators value proposition with edge computing." Technical Brief.
Okay, F. Y. and S. Ozdemir (2016). A fog computing based smart grid model. 2016 international symposium on networks, computers and communications (ISNCC), IEEE.
Oueis, J., E. C. Strinati and S. Barbarossa (2015). The fog balancing: Load distribution for small cell cloud computing. 2015 IEEE 81st vehicular technology conference (VTC spring), IEEE.
Pan, Y., P. Thulasiraman and Y. Wang (2018). Overview of cloudlet, fog computing, edge computing, and dew computing. Proceedings of The 3rd International Workshop on Dew Computing.
Pareek, K., P. K. Tiwari and V. Bhatnagar (2021). Fog Computing in Healthcare: A Review. IOP Conference Series: Materials Science and Engineering, IOP Publishing.

Patwary, A. A.-N., R. K. Naha, S. Garg, S. K. Battula, M. A. K. Patwary, E. Aghasian, M. B. Amin, A. Mahanti and M. Gong (2021). "Towards secure fog computing: A survey on trust management, privacy, authentication, threats and access control." Electronics **10**(10): 1171.
Pavlovic, D. (2008). Network as a computer: ranking paths to find flows. International Computer Science Symposium in Russia, Springer.
Peng, Y., Y. Zhao and J. Hu (2023). "On the role of community structure in evolution of opinion formation: A new bounded confidence opinion dynamics." Information Sciences **621**: 672-690.
Prudencio, R. F., M. R. Maximo and E. L. Colombini (2023). "A survey on offline reinforcement learning: Taxonomy, review, and open problems." IEEE Transactions on Neural Networks and Learning Systems.
Puliafito, C., C. Vallati, E. Mingozzi, G. Merlino and F. Longo (2021). "Design and evaluation of a fog platform supporting device mobility through container migration." Pervasive and Mobile Computing **74**: 101415.
Puthal, D., M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty and A. Y. Zomaya (2018). "Secure and sustainable load balancing of edge data centers in fog computing." IEEE Communications Magazine **56**(5): 60-65.
Qu, Z., X. Liu and M. Zheng (2022). "Temporal-Spatial Quantum Graph Convolutional Neural Network Based on Schrödinger Approach for Traffic Congestion Prediction." IEEE Transactions on Intelligent Transportation Systems.
Rafique, H., M. A. Shah, S. U. Islam, T. Maqsood, S. Khan and C. Maple (2019). "A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing." IEEE Access **7**: 115760-115773.
Ray, S. (2019). A quick review of machine learning algorithms. 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon), IEEE.
Refaat, H. E. and M. A. Mead (2019). "DLBS: Decentralize Load-Balance Scheduling Algorithm for Real-Time IoT Services in Mist Computing." Editorial Preface From the Desk of Managing Editor… **10**(9).
Refaat, H. E. and M. A. Mead (2019). "DLBS: decentralize load-balance scheduling algorithm for real-time IoT services in mist computing." International Journal of Advanced Computer Science and Applications **10**(9).
Roig, P. J., S. Alcaraz, K. Gilly and C. Juiz (2020). Modelling a leaf and spine topology for VM migration in Fog computing. 2020 24th International Conference Electronics, IEEE.
Ruan, L., S. Guo, X. Qiu and R. Buyya (2021). Fog computing for smart grids: challenges and solutions. Electric Vehicle Integration in a Smart Microgrid Environment, CRC Press**:** 7-31.
Sabireen, H. and V. Neelanarayanan (2021). "A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges." ICT Express **7**(2): 162-176.
Sadeeq, M. M., N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami and R. R. Zebari (2021). "IoT and Cloud computing issues, challenges and opportunities: A review." Qubahan Academic Journal **1**(2): 1-7.
Salaht, F. A., F. Desprez and A. Lebre (2020). "An overview of service placement problem in fog and edge computing." ACM Computing Surveys (CSUR) **53**(3): 1-35.
Salimian (2021). "Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment." Software: Practice and Experience.

Salimian, M., M. Ghobaei-Arani and A. Shahidinejad (2021). "Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment." Software: Practice and Experience.
Sami (2021). "Demand-Driven Deep Reinforcement Learning for Scalable Fog and Service Placement." IEEE Transactions on Services Computing.
Sami, H., A. Mourad, H. Otrok and J. Bentahar (2021). "Demand-Driven Deep Reinforcement Learning for Scalable Fog and Service Placement." IEEE Transactions on Services Computing.
Sarma, B., G. Kumar, R. Kumar and T. Tuithung (2019). Fog Computing: An Enhanced Performance Analysis Emulation Framework for IoT with Load Balancing Smart Gateway Architecture. 2019 International Conference on Communication and Electronics Systems (ICCES), IEEE.
Satka, Z., M. Ashjaei, H. Fotouhi, M. Daneshtalab, M. Sjödin and S. Mubeen (2023). "A comprehensive systematic review of integration of time sensitive networking and 5G communication." Journal of Systems Architecture: 102852.
Sethi, V. and S. Pal (2023). "FedDOVe: A Federated Deep Q-learning-based Offloading for Vehicular fog computing." Future Generation Computer Systems **141**: 96-105.
Shakarami, A., H. Shakarami, M. Ghobaei-Arani, E. Nikougoftar and M. Faraji-Mehmandar (2022). "Resource provisioning in edge/fog computing: A Comprehensive and Systematic Review." Journal of Systems Architecture **122**: 102362.
Shakeel, H. and M. Alam (2022). "Load Balancing Approaches in Cloud and Fog Computing Environments: A Framework, Classification, and Systematic Review." International Journal of Cloud Applications and Computing (IJCAC) **12**(1): 1-24.
Sheikh Sofla, M., M. Haghi Kashani, E. Mahdipour and R. Faghih Mirzaee (2022). "Towards effective offloading mechanisms in fog computing." Multimedia Tools and Applications **81**(2): 1997-2042.
Shi, W., G. Pallis and Z. Xu (2019). "Edge computing [scanning the issue]." Proceedings of the IEEE **107**(8): 1474-1481.
Singh, S. P. (2022). "Effective Load Balancing Strategy Using Fuzzy Golden Eagle Optimization in Fog Computing Environment." Sustainable Computing: Informatics and Systems: 100766.
Sofla, M. S., M. H. Kashani, E. Mahdipour and R. F. Mirzaee (2022). "Towards effective offloading mechanisms in fog computing." Multimedia Tools and Applications: 1.
Souza, V. B. C., W. Ramírez, X. Masip-Bruin, E. Marín-Tordera, G. Ren and G. Tashakor (2016). Handling service allocation in combined fog-cloud scenarios. 2016 IEEE international conference on communications (ICC), IEEE.
Statista. (2023). "IoT connected devices by vertical 2030 ", from https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/.
Stojmenovic, I. and S. Wen (2014). The fog computing paradigm: Scenarios and security issues. 2014 federated conference on computer science and information systems, IEEE.
Sulimani, H., W. Y. Alghamdi, T. Jan, G. Bharathy and M. Prasad (2021). "Sustainability of Load Balancing Techniques in Fog Computing Environment." Procedia Computer Science **191**: 93-101.
Sulimani, H., A. M. Sajjad, W. Y. Alghamdi, O. Kaiwartya, T. Jan, S. Simoff and M. Prasad (2022). "Reinforcement optimization for decentralized service placement policy in IoT-centric fog environment." Transactions on Emerging Telecommunications Technologies: e4650.

Taneja (2017). Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE.
Taneja, M. and A. Davy (2017). Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), IEEE.
Tang, Q., R. Xie, F. R. Yu, T. Huang and Y. Liu (2020). "Decentralized computation offloading in IoT fog computing system with energy harvesting: A dec-POMDP approach." IEEE Internet of Things Journal **7**(6): 4898-4911.
Tarawneh, A. and A. Hyasat (2010). "White Spot Formation under Banded Molars after Orthodontic Treatment and Suggested Preventive Measures." Journal of the Royal Medical Services **17**(3): 45.
Téllez, N., M. Jimeno, A. Salazar and E. Nino-Ruiz (2018). "A tabu search method for load balancing in fog computing." Int. J. Artif. Intell **16**(2).
Téllez, N., M. Jimeno, A. Salazar and E. Nino-Ruiz (2018). "A tabu search method for load balancing in fog computing." Int. J. Artif. Intell **16**(2): 1-30.
Tiwary, M., S. Sharma, P. Mishra, H. El-Sayed, M. Prasad and D. Puthal (2018). Building Scalable Mobile Edge Computing by Enhancing Quality of Services. 2018 International Conference on Innovations in Information Technology (IIT), IEEE.
Tran-Dang, H. and D.-S. Kim (2023). Bandit Learning for Distributed Task Offloading in Fog Computing Networks: Literature Review, Challenges, and Open Research Issues. International Conference on Network-Based Information Systems, Springer.
Tran-Dang, H. and D.-S. Kim (2023). "Dynamic collaborative task offloading for delay minimization in the heterogeneous fog computing systems." Journal of Communications and Networks.
Traub, M., A. Maier and K. L. Barbehön (2017). "Future automotive architecture and the impact of IT trends." IEEE Software **34**(3): 27-32.
Vaquero, L. M. and L. Rodero-Merino (2014). "Finding your way in the fog: Towards a comprehensive definition of fog computing." ACM SIGCOMM computer communication Review **44**(5): 27-32.
Velasquez, K., D. P. Abreu, M. Curado and E. Monteiro (2021). "Service Placement for Latency Reduction in the Fog using Application Profiles." IEEE Access.
Verma, M., N. Bhardwaj and A. K. Yadav (2016). "Real time efficient scheduling algorithm for load balancing in fog computing environment." Int. J. Inf. Technol. Comput. Sci **8**(4): 1-10.
Verma, S., A. K. Yadav, D. Motwani, R. Raw and H. K. Singh (2016). An efficient data replication and load balancing technique for fog computing environment. 2016 3rd international conference on computing for sustainable global development (INDIACom), IEEE.
Vögler, M., J. M. Schleicher, C. Inzinger and S. Dustdar (2016). "A scalable framework for provisioning large-scale IoT deployments." ACM Transactions on Internet Technology (TOIT) **16**(2): 1-20.
Wang (2017). "Online placement of multi-component applications in edge computing environments." IEEE Access **5**: 2514-2533.
Wang, B., Y. Zhang and W. Zhang (2022). "A Composite Adaptive Fault-Tolerant Attitude Control for a Quadrotor UAV with Multiple Uncertainties." Journal of Systems Science and Complexity **35**(1): 81-104.
Wang, S., M. Zafer and K. K. Leung (2017). "Online placement of multi-component applications in edge computing environments." IEEE Access **5**: 2514-2533.

Wang, Y., X. Han and S. Jin (2023). "MAP based modeling method and performance study of a task offloading scheme with time-correlated traffic and VM repair in MEC systems." Wireless Networks **29**(1): 47-68.
Wójcicki, K., M. Biegańska, B. Paliwoda and J. Górna (2022). "Internet of Things in Industry: Research Profiling, Application, Challenges and Opportunities—A Review." Energies **15**(5): 1806.
Wu, H., S. Jin and W. Yue (2022). "Pricing policy for a dynamic spectrum allocation scheme with batch requests and impatient packets in cognitive radio networks." Journal of Systems Science and Systems Engineering **31**(2): 133-149.
Wu, Q., H. Liu, R. Wang, P. Fan, Q. Fan and Z. Li (2019). "Delay-sensitive task offloading in the 802.11 p-based vehicular fog computing systems." IEEE Internet of Things Journal **7**(1): 773-785.
Xiao, Z., J. Shu, H. Jiang, J. C. Lui, G. Min, J. Liu and S. Dustdar (2022). "Multi-objective parallel task offloading and content caching in D2D-aided MEC networks." IEEE Transactions on Mobile Computing.
Xu, D., L. Liu, N. Zhang, M. Dong, V. C. Leung and J. A. Ritcey (2023). "Nested Hash Access with Post Quantum Encryption for Mission-Critical IoT Communications." IEEE Internet of Things Journal.
Xu, X., S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun and A. X. Liu (2018). "Dynamic resource allocation for load balancing in fog environment." Wireless Communications and Mobile Computing **2018**.
Yang (2015). "Cost aware service placement and load dispatching in mobile cloud systems." IEEE Transactions on Computers **65**(5): 1440-1452.
Yang, D., T. Zhu, S. Wang, S. Wang and Z. Xiong (2022). "LFRSNet: A robust light field semantic segmentation network combining contextual and geometric features." Frontiers in environmental Science **10**: 996513.
Yannuzzi, M., R. Irons-Mclean, F. van Lingen, S. Raghav, A. Somaraju, C. Byers, T. Zhang, A. Jain, J. Curado and D. Carrera (2017). Toward a converged openfog and etsi mano architecture. 2017 IEEE Fog World Congress (FWC), IEEE.
Yi, S., Z. Qin and Q. Li (2015). Security and privacy issues of fog computing: A survey. Wireless Algorithms, Systems, and Applications: 10th International Conference, WASA 2015, Qufu, China, August 10-12, 2015, Proceedings 10, Springer.
Zahoor, S., S. Javaid, N. Javaid, M. Ashraf, F. Ishmanov and M. K. Afzal (2018). "Cloud–fog–based smart grid model for efficient resource management." Sustainability **10**(6): 2079.
Zeng (2016). "Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system." IEEE Transactions on Computers **65**(12): 3702-3712.
Zhang, X., Z. Wang and Z. Lu (2022). "Multi-objective load dispatch for microgrid with electric vehicles using modified gravitational search and particle swarm optimization algorithm." Applied Energy **306**: 118018.
Zhang, X., S. Wen, L. Yan, J. Feng and Y. Xia (2022). "A hybrid-convolution spatial–temporal recurrent network for traffic flow prediction." The Computer Journal: bxac171.

APPENDIX

Appendix I: COMPARISON OF RELATED WORKS.

Authors	Contributions	Offloading-in	Offloading Approach	Clustering	TSA	Control System	Metrics	Pros/Cons
Jiang, Chen et al. (2018)	Offloading dispatcher and an energy-efficient offloading decision mechanism	Fog nodes	Find the optimum destination device after studding the offloading cost- Dynamic	No	Yes	Decentralized	Response time, Energy consumption	The algorithm costs the system a high number of exchanged messages to explore the suitable device for offloading.
Ebrahim and Hafid (2023)	A LB algorithm based on Reinforcement Learning (RL), DDQN.	Cloud and Fog Servers	The algorithm aims to minimize the waiting delay of IoT workloads in dynamic environments with unpredictable traffic demands, using intelligent workload distribution.	Yes	No	Distributed	Latency, Waiting time, Execution time, Response time	The author does not solve the inherent issues of traditional offloading.
Albalawi, Alkayal et al. (2022)	PSOSVR: based on a many-objective Particle Swarm Optimization (PSO) algorithm with Support Vector Regression (SVR)	Fog nodes	Dynamic- AI	No	No	Centralized- The control unit locates in Fog layer - FSM	Response time, Energy consumption, Resource utilization, and Throughput	The authors have not given an advantage in execution for time-sensitive applications or distant offloading. The architecture has low scalability.
Lu, Gu et al. (2020)	DRL: based on the improved IDRQN algorithm.	Cloud and Fog Servers	DRL is proposed to solve the offloading problem of multiple service nodes for the cluster and multiple dependencies for mobile tasks in large-scale heterogeneous MEC.	Yes	Yes	Decentralized	Energy consumption, load balancing, latency and average execution	The study considers TSA and distant offloading. However, it follows the traditional offloading approach, which may consume unnoticeable time.

Tran-Dang and Kim (2023)	Proposing DCTO, a dynamic collaborative task offloading, algorithm.	Fog nodes	Using partitioned tasks and parallel computation	No	No	Decentralized	Average of task execution delay and utilization ratio of fogs	The authors did not give advantage for sensitive applications or distant offloading.
Gowri and Baranidharan (2023)	A dynamic energy resource allotment (DERA) technique	Cloud and Fog Servers	The proposed algorithm used two algorithms to find the optimum target device.	No	No	Centralized- The control unit locates in Fog layer - Controller	Broadband costs, duration, and energy consumption.	The algorithm shows outstanding results among others. However, it ignores sensitive application and distant offloading.
Hussein and Mousa (2020)	Two nature-inspired meta-heuristic schedulers, namely ant colony optimization (ACO) and particle swarm optimization (PSO),	Cloud and Fog Servers	The proposed model considers the network latency and the service rate of the fog nodes	No	No	Centralized- The control unit locates in Fog layer –Fog Master Node	Communication cost and Response time	The authors do not prioritize the critical application to be executed locally. Moreover, they ignore other critical problems.
Li, Zhuang et al. (2018)	A self-similarity-based load balancing (SSLB) mechanism for large-scale fog computing.	Fog nodes	address the LB challenges caused by fog's 'large-scale' characteristic through clustering.	Yes	Yes	Decentralized	Execution time, clustering overhead	Even though SSLB presents many features, it faces the inherent issues of prevalent offloading, such as decision time and messages.
Lu, Wu et al. (2023)	Two scenarios are considered: Unlimited-Processor Fog Nodes (UPFN) and Limited-Processor Fog Nodes (LPFN).	Cloud and Fog Servers	minimize the total monetary cost by considering the deadline. and capacity constraints.	No	Yes	fog nodes' distributions are concentrated	Average cost and Makespan	The manuscript does not find a solution for large-scale networks and other phenomena in dynamic offloading.

Sarma, Kumar et al. (2019)	A smart gateway as a load balancer in a fog environment	Cloud and Fog Servers	The authors proposed a smart gateway that control the arrival tasks with minimum cost.	No	No	Centralized- - The control unit locates in Fog layer –Smart gateway	Network delay and Computing time	Despite the outcomes of the centralized system, the proposed solution did not consider other inherent issues of offloading costs.
Chakraborty and Mazumdar (2023)	Hybrid metaheuristic Greedy Randomized Adaptive Search Procedure and Genetic Algorithm (GRASP-GA).	Cloud and Fog Servers	Hybrid metaheuristic == > Dynamic- AI. A dynamic edge server selection mechanism for task offloading.	Yes	Yes	Centralized. The control unit locates in cloud layer.	total execution time and energy consumption	The authors did not study the implications of distant offloading on critical applications. The proposed model highly uses a cloud server. Even though its outstanding results, assigning sensitive applications to the fog devices is a better approach.
HybOff	A comprehensive offloading algorithm that override most of inherent issues in static and dynamic offloading.	Fog layer	The proposed algorithm aims to utilize many features, such as clustering, static offloading, and local execution, to present a comprehensive solution.	Yes	Yes	Central-distributed	RU, LB level, and system performance	It solves many inherent issues in prevalent offloading, such as offloading decision, distant offloading, network