Efficient Edge-Cloud Resource Management for Latency-Sensitive Applications

by

Jaber Faraj J Almutairi

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

UNIVERSITY OF LEEDS

The University of Leeds

School of Computing

April 2020

Declaration

The candidate confirms that the work submitted is his own, except where work which has formed part of jointly-authored publications has been included. The contribution of the candidate and the other authors to this work has been explicitly indicated below. The candidate confirms that appropriate credit has been given within the thesis where reference has been made to the work of others.

- McKee, D. W., Clement, S. **J., Almutairi**, J., & Xu, J. (2017, March). Massive-scale automation in cyber-physical systems: vision & challenges. In 2017 IEEE 13th International Symposium on Autonomous Decentralized System (ISADS) (pp. 5-11). IEEE. My contribution in this jointly authored publication was to understand Edge Computing Environment. I literature review about Edge computing and Internet of things applications. The content of this paper is included throughout the thesis and mainly in Chapter 2.

- McKee, D. W., Clement, S. **J., Almutairi**, J., & Xu, J. (2018). Survey of advances and challenges in intelligent autonomy for distributed cyber-physical systems. CAAI Transactions on Intelligence Technology, 3(2), 75-82. My contribution in this jointly authored publication was to list and identify current research issues and challenges in Edge computing area. This was one of the first steps toward doing my PhD. The content of this paper is included throughout the thesis and mainly in Chapter 2.

- **Almutairi, Jaber**, and Jie Xu. "Investigation on Workload Variations in an Edge Computing Environment." 35th UK Performance Engineering Workshop. Newcastle University, 2018. This paper is the candidate’s own work. It was reviewed by the co-author Prof. Jie Xu. The content of this paper is included throughout the thesis and mainly in Chapter 3.

Dedication

To my father, Abo Jaber, who is always inspirational throughout my life. His unlimited support and encouragement helped me a lot to reach the level where I am now.

To my mother, who is like a candle that consumes itself to light the way for us, with love and passion. Without your prayers and assistance, I would not have reached my goals.

To my wife and my kids Sarah and Faisal. It is because of your patience, love and support, I kept upholding to my aims.

Acknowledgements

First and foremost, all praise and thanks to Allah The Almighty for His graces and guidance for giving me the strength, well-being and ability to undertake some of the most important work and experience in my life.

My deep thanks go to my Supervisors Prof. Jie Xu and Dr. Paul Townend for their guidance, support and advice throughout my studies at the University of Leeds. Additionally, I would like to thank my thesis examiners Prof. Karim Djemame, Dr. Graham Morgan and Dr. Zheng Wang for their excellent comments and feedback when examining this Thesis.

I would also like to thank the DSS group members for the great meetings and seminars we used to enjoy almost every week. Also, my thanks go to people in the School of Computing, I have always been proud of being one of its members.

Finally, I would never have the chance to study in the UK without the financial aid of my sponsor, Taibah University and the government of the Kingdom of Saudi Arabia. I hope to be able to repay part of the debt by transferring knowledge and promoting research in Saudi Arabia.

Abstract

Internet of Things (IoT) is quickly evolving into a disruptive technology in recent years. For enhancing customer experience and accelerating job execution, IoT task offloading enables mobile end devices to release heavy computation and storage to the resource-rich nodes in collaborative Edges or Clouds. Resource management at the Edge-Cloud environment is challenging because it deals with several complex factors (e.g. different characteristics of IoT applications and heterogeneity of resources). Thus, efficient resource management will play an essential role in providing real-time or near real-time use for IoT applications. However, how different service architecture and offloading strategies quantitatively impact the end-to-end service time performance of IoT applications is still far from known particularly given a dynamic and unpredictable assortment of interconnected virtual and physical devices.

This PhD thesis has investigated and modelled the delay within the Edge-Cloud environment as well as providing a detailed analysis of the main factors of service latency. Moreover, proposing a new task offloading approach for latency-sensitivity applications using fuzzy logic, where a decision is made as to whether we can offload the task to Local Edge, other Collaborative Edge or the Cloud depending on the current parameters of both application characteristics and the resources within the Edge-Cloud Environment. The proposed approach was compared against existing related works using a simulation tool, and it was evaluated in the domain of the edge-cloud environment where it was found to improve the overall service time for latency-sensitive applications, effectively utilising the edge-cloud resources.

List of Abbreviations

IoT	Internet of Things
AR	Augmented Reality
QoS	Quality of Service
CPU	Central Processing Unit
VM	Virtual Machine
PC	Personal Computer
MCC	Mobile Cloud Computing
IoE	Internet of Everythings
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
GPS	Global Positioning System
LAN	Local Area Network
WAN	Wide Area Network
MAN	Metropolitan Area Network
WLAN	Wireless Local Area Network
RAN	Radio Access Network
VR	Virtual Reality
CSV	Comma-Separated Values
MI	Million Instructions
CCTV	Closed-Circuit television
EC	Edge Controller
MIPS	Million Instructions Per Second

MAPE	Monitor, Analyse, Plan and Execute
GPU	Graphics Processing Unit
FPGA	Field-Programmable Gate Array

Table of Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Motivation

Nowadays, IT sector has developed at a massive rate: with more than 50 billion devices will be connected to the internet in the coming years [1] which known as the internet of things (IoT) era [2]; with a tremendous amount of streaming data gathered by IoT devices, needs to be transferred, processed, and stored; with various applications domains such as autonomous vehicles, Augmented Reality (AR), online video games, smart city Industry 4.0 etc. Hence, this immense growth requires platforms to support the increased amount of IoT devices as well as organise and process the produced data since IoT devices are limited in term of power and computational capabilities, i.e. CPU and memory [3]. This primarily affects the adoption of compute-intensive applications such as (AR), Online Gaming and processing of video streaming [4].

Cloud Computing is one of the main factors to support this growth by enables on-demand access to a massive pool of computation resources for services process and data analytics [5]. However, since the Cloud is far away form IoT devices, also there is an enormous amount of generated data needs to be transferred and processed in a real-time manner. Consequently, applications that require low-latency, real-time interaction and high Quality of Service (QoS) suffer from the Cloud due to network delay [6]. Further, data is increasingly produced at the edge of the network, hence, it would be more efficient to also process the data at the edge level because cloud computing is not always practical for data processing when the data is produced at the edge and require processing in a real-time manner.

Therefore, the concept of Edge computing has appeared to complement Cloud services. It basically refers to an intermediate layer with computation capabilities between the Cloud and closer to IoT devices to fill latency gaps. Edge computing provides the opportunity to serve better streaming services, which is both latency-sensitive and bandwidth-intensive such as Google Stadia and Netflix. It allows avoiding the Uploading /download of massive files as well as the pre-processing of offloading tasks, which contribute to minimising the overall service time.

Although Edge computing is a promising enabler for latency-sensitive applications because of the closeness to IoT devices. Computational resources in edge computing similar to cloud computing which consist of a pool of servers that operated and managed virtually as well as hosted at the edge of the network. Efficient Edge-Cloud resource management for latency-sensitive applications is essential to fully utilise the capabilities of Edge nodes [7]. However, latency-sensitive applications have various changing characteristics, such as computational demand and communication demand. Consequently, the latency depends on the scheduling policy of applications offloading tasks as well as where the jobs will be placed. Therefore, Edge-Cloud resource management should consider these characteristics in order to meet the requirements of latency-sensitive applications. On the other hand, another challenge for improving the utilisation the edge-cloud resources and reducing the dependency of the cloud, which will help to reduce the overall cost.

A number of studies have been conducted on edge cloud resource management to achieve improvements in the overall service time for IoT applications. A few studies have a particular focus on latency-sensitive applications, and others have focused on resource utilisation and energy efficiency as the main objectives. Hence, there is a need for holistic and efficient resource management that considers characteristics of offloading IoT applications' tasks (computation, communication and latency), as well as resource parameters, such as resource utilisation and resource heterogeneity in order to meet the required service time for the applications and utilising Edge-Cloud resources efficiently.

Thus, it is essential to conduct in-depth research to investigate the latency within the edge-cloud system, the impact of computation and communication demands and resource heterogeneity to provide a better understanding of the problem and facilitate the development of an approach that aims to improve both applications’ QoS and edge-cloud system performance.

1.2 Aims and Objectives

Offloading tasks of IoT latency-sensitive applications have diverse demands in terms of computation and communication; for example, some tasks require intense computation and a small amount of transferred data, while others do not. Thus, varying demands will affect the offloading decision as well as the overall service time. In addition, the host server in the edge plays an essential role in the processing service time (i.e., offloading a task to an overloaded edge server will incur long latency). This research has two aims: one is to improve the overall service time of latency-sensitive applications considering the varying of computational and communication demands, and the other is to utilise edge-cloud resources with a consideration of resource heterogeneity.

These aims require an in-depth analysis and investigation. Thus, the following research questions need to be addressed:

- **Q1**: How can the latency of the service time based on the offloading site in the edge-cloud environments be modelled, and what will be the impact on computation and communication time?
- **Q2:** How does the task variation in terms of computation demands and communication demands affect the offloading decision, and what is the impact on the overall service time for latency-sensitive applications?
- **Q3:** How does the resource heterogeneity in edge-cloud environments affect the offloading decision, and what is the impact on the overall service time for latency-sensitive applications?
- **Q4:** How do different offloading strategies quantitatively influence the end-to-end service time performance of IoT applications and services?

Specifically, the main objectives of this study are:

1- Exploring the scheduling of tasks offloading issues and challenges in the edge-cloud paradigm. Task offloading mechanisms of latency-sensitive applications has been an active research area, especially with the trade-off of application QoS, such as service time and resource utilisation. Therefore, it is necessary to have an in-depth understanding of the current issues in order to propose an approach that can be used to address these issues.

2- Investigating the parameters that influence the overall service time in edge-cloud environments. It is necessary to investigate the impact of the location of the offloaded tasks, as well as the effect of task variation of latency-sensitive applications in terms of the requirements of computation and communication and the impact of resource heterogeneity in edge-cloud environments and how it affects overall service time.

3- Developing a dedicated approach for offloading tasks to handle the requirements of latency-sensitive IoT applications and efficiently utilising the resources in edge-cloud environments in order to minimise the overall service time. Currently, there are a number of offloading algorithms in edge-cloud environments, but they are mainly focussed on either applications' characteristics or edge-cloud resource utilisation. Thus, there is a need for an approach that has a consideration of both, which is the objective of this work.

1.3 Methodology

Resource management in edge computing is a complicated process as it involves managing a diversity of resources, such as edge nodes and the central cloud, to achieve the computational requirements of end devices. It should consider the constraints in terms of computational capacity and network bandwidth of any type of resources as well as the demands of end-device applications. Therefore, there are three scientific research methods used to research the edge computing environment that demonstrate and investigate latency-sensitive IoT applications with their intensive data, resources in edge nodes and the cloud.

a. Direct Experiments

In the context of this research, this approach can be described as conducting a direct experiment that implements an edge computing environment to validate a hypothesis or implement an idea for real-ward examples, which can increase result reliability and accuracy. However, this approach is generally time-consuming and involves several constraints, such as resource accessibility and availability. Moreover, in edge computing environments, setting up a testbed that provides virtualised resources (computational and network) can be quite expensive[8] [9] and may present challenges for conducting repeatable and scalable experiments for a comprehensive performance analysis [10].

b. Mathematical Modelling

This method can be defined as formulating the system or the environment using mathematical models subject to a set of assumptions to provide a more comprehensive understanding of a system's components. The outcomes of this method can be validated through direct experiments or reliable simulator tools.

c. Simulation

Simulation is another scientific method that can be used to test a hypothesis or provide a solution. With simulator tools, experiments can be conducted in reasonable time and support the repeatability for required modifications. Due to the heterogeneity of resources and the variety of IoT applications, simulator tools are commonly used in the area of edge computing [10].

In this research, both mathematical modelling and simulation methods are used. Mathematical modelling is used to formulate latency models in offloading decisions in edge-cloud environments, and EdgeCloudSim [11] is used as an edge-cloud simulator that can fit with the context of the study (more details presented in section 2.7).

In general, a simulation-based method can be used in scientific research when one of the following conditions is met [12]:

1- A complete environment to conduct the research does not exist.
2- The process of a direct experiment is complex because it involves systems with many influencing parameters, many dependency factors, and a huge amount of data, thus tending to make the experiment intricate and unmanageable.
3- The research requires a long time frame. The simulation-based method can compress the time frame so that the research can be conducted within the required time, for example, by accelerating the analysis process.
4- The configurations and setups of a direct experiment are difficult and time-consuming. Thus, the experiment is not repeatable or reusable.

A simulation-based method can be used for several research purposes such as evaluation, comparison, sensitivity analysis and optimization [12]:

1- Evaluation: examining the proposed approach with specific criteria

2- Comparison: allowing the researchers to compare their methods with those of other competitors in the same field
3- Sensitivity analysis: determining the impact of several factors on system performance
4- Optimization: identifying the trade-offs between the combination of parameters and its effect on the system performance

In this research, the simulation tool was selected to avoid significant research challenges in the area of edge computing resource management. There are two main reasons why the simulation tool was used in this research. First, similar to with research in cloud computing, service providers do not allow third parties to access their infrastructure to get the necessary data for research purposes [13]. Second, setting up a real Edge-Cloud testbed is both costly, and time-consuming [14]. Therefore, most of the related studies in the field are simulation-based and/or theoretical.

Several issues should be considered in order to select the appropriate simulation tools and validate the proposed approach and its results. This will help to overcome and mitigate the general limitations of the simulation-based method. These issues are the following:

1- <u>The selection of the appropriate simulator tool:</u> According to [], the first step is to search for the simulator tools that have the necessary functionalities to deliver the research aims and objectives. Other criteria, such as (the required HW to run the simulation, the efficiency of generated logs files, the cost of simulation and assistance availability), could also be used to select the simulation tool [15]. In this research, we reviewed a collection of the existing tools (e.g. IfogSim, IoTSim, and FogNetSim++), comparing their main focuses. EdgeCloudSim was the most appropriate for the purpose of this research. More details are in section 2.7.

2- The selection of simulation parameters: In a simulation-based method, the generated result depends on the selected parameters, and therefore, any wrong parameters could lead to incorrect results. In this research, we selected the parameters based on related research in both application demands (e.g. computational and communications) and edge-cloud resources (e.g. VM configuration). Additionally, a sensitivity analysis was conducted to investigate the impact of the application parameters.

3- The validation of the generated results: In order to avoid any anomalies from the simulation results, each simulation was run five times, and a statistical analysis was conducted to take the mean values. Then, some of the results obtained were compared with the general findings from other research. For example, some of the generated results were consistent with other researchers' results from direct experiments.

1.4 Main Contributions

The main contributions of this work can be summarised as follows:

- Presenting a model that can show the impact of different tasks' offloading scenarios for time-sensitive applications in terms of end-to-end service times. It provides in-depth analyses of the offloading latency models that consider computation and communication as key parameters with respect to offloading to the local edge node, other edge nodes or the cloud.

- Quantifying the impact of the variations of the offloading tasks and the performance of different computational resources within the edge-cloud system. Different computation and communication demands of offloading tasks, as well as different VMs, have been modelled in the simulation tool, which has helped to quantify the impact of computation and communication demands of offloading tasks.
- Proposed a new approach that adopts the fuzzy logic algorithm which considers application characteristics (e.g., CPU demand, network demand and delay sensitivity) as well as resource utilisation and resource heterogeneity in order to minimise the overall time of latency-sensitive applications.

1.5 Thesis Overview

The remaining chapters of this thesis are organised as follows:

- **Chapter 2** presents an overview of the fundamental concepts of the subject of scheduling offloading tasks un the edge-cloud system. Firstly, the core concepts of cloud computing with more details on its definition, architecture, deployment models and the idea of mobile cloud computing as an extended model for cloud computing will be presented. Secondly, the core concepts of the transformation to the edge computing and its models will be discussed. These presented the idea of edge computing and explain the different terms such as fog computing, mobile edge computing, etc., with a comparison between them. Also, the concept of the internet of things (IoT) and its applications are described. After that, the concept of offloading tasks is introduced and discussed with the context of edge computing. This is followed by positioning the work in the related literature, focusing on the scheduling offloading tasks issues and resource management in Edge-Cloud system. A reviewing with related works that focus on application characteristics is presented. Also, the related works that consider parameters of edge cloud resources such as resource utilisation and resource heterogeneity is provided. Finally, research open challenges and simulation tools are presented.

- **Chapter 3** presents the overview of the edge-cloud system architecture that supports scheduling offloading tasks of IoT applications, as well as the explanation of the required components and their interactions within the system architecture. Furthermore, it presents the offloading latency models that consider computation and communication as crucial parameters with respect to offloading to the local edge node, other edge nodes or the cloud. Chapter 3 concludes by discussing early experiments conducted on EdgeCloudSim to investigate and evaluate the latency models of each offloading scheme.
- **Chapter 4** presents and discusses the main factors of service latency that will be considered in the proposed approach for edge-cloud resource management. Since the demand for computation and communication tasks vary in IoT applications, this chapter aims to validate the impact of these factors on the overall application latency.

Moreover, Edge-Cloud environment consists of heterogeneity of computing resources; thus, selecting the appropriate resources to process the offloading tasks play a critical role to improve the overall service time. Therefore, a number of simulation experiments were set up to evaluate the influence of these factors.

- **Chapter 5** proposes a new approach for task offloading in edge-cloud systems in order to minimise the overall service time for latency-sensitive applications. The approach adopts the fuzzy logic algorithm that considers application characteristics (e.g., CPU demand, network demand and delay sensitivity) as well as resource utilisation. A number of simulation experiments have been conducted in order to evaluate the proposed approach with other related work.
- **Chapter 6** presents a summary of the work and contributions presented in this thesis. Moreover, discuss an overall evaluation of thesis objectives and provides some of the potential topics for future work that could further enhance this research.

Chapter 2. Challenges and Existing Work in Edge-Cloud Systems

2.1 Overview

This chapter describes the essential background concepts of this research - i.e. improving the overall service time performance for latency-sensitive applications as well as manage the resource efficiently in Edge-Cloud environments. It starts by presenting the basic background of Cloud Computing and Mobile Cloud computing with a detailed description of its definition, system architecture, services types and deployment, as shown in Section 2.2. The following section presents the development of Cloud Computing paradigm by the concepts of Edge computing with a detailed description of its, definition, models and related technologies. Finally, the concepts of offloading tasks are presented with the state-of-the-art methods that aim to efficiently manage the resource of the Edge-Cloud and enhance the overall service time. It concludes with a description of the research method.

2.2 Cloud Computing

2.2.1 Definition

Cloud computing is a computing model that has characteristics to support the services of IoT and applications of Big Data. It is defined as *"A model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management, effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models"*[5].

2.2.2 Characteristics

Gong et al. (2010) introduced the main characteristics of cloud computing, being on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service [16].

1. **On-demand self-service**: Cloud computing allows consumers to provide the service to them without human interaction at any time.
2. **Broad network access**: Services are available over the network and accessed through standard mechanisms that support a variety of platforms, such as mobile phones, PCs, etc. This is called a service-oriented architecture model that permits the components of the cloud to be available over the network as a service.
3. **Resource pooling**: Variety of resources that can serve many consumers with different demands at the same time.

4. **Rapid elasticity**: Customers may expand their usage as much as they choose. It seems to the customers that the services are unlimited, and they can receive any quantity at any time.
5. **Measured services**: All services in the cloud, such as storage and processing data, are measured automatically.

2.2.3 Service Types

Cloud computing has three service models software as a service (SaaS), platform as a service (PaaS) model and infrastructure as a service (IaaS) [17], see figure 2.1.

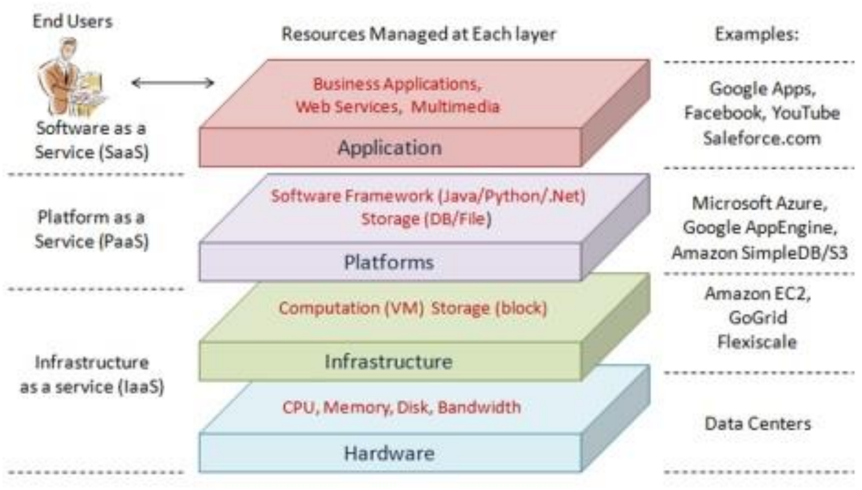

End Users
Resources Managed at Each layer
Examples:
Software as a Service (SaaS)
Business Applications, Web Services, Multimedia
Application
Google Apps, Facebook, YouTube Saleforce.com
Platform as a Service (PaaS)
Software Framework (Java/Python/.Net) Storage (DB/File)
Platforms
Microsoft Azure, Google AppEngine, Amazon SimpleDB/S3
Computation (VM) Storage (block)
Infrastructure
Amazon EC2, GoGrid Flexiscale
Infrastructure as a service (IaaS)
CPU, Memory, Disk, Bandwidth
Hardware
Data Centers

Figure 2. 1: Cloud Computing services model and examples [18]

1. **Software as a service (SaaS):** The provider of cloud computing services allows the customer to use applications in the cloud, such as Microsoft Word Online. The applications work with heterogeneous

platforms. The customers do not have the right to manage the cloud infrastructure, such as the server, storage or network.

2. **Platform as a service (PaaS):** The provider of the cloud computing services permits consumers to create their applications using programming languages, libraries, services and tools supported by the provider. The customers do not have the rights to manage the cloud infrastructure, such as the server, storage or network.
3. **Infrastructure as a service (IaaS):** The provider of the cloud service allows the customers to have control in their own deployed applications, storage and operating system. The customers do not have the right to manage the cloud infrastructure, such as a server, storage or network, but may have limited control of types of network components.

2.2.4 Deployment Types

According to [16], cloud computing has four deployment models - private cloud, community cloud, public cloud and hybrid cloud.

1. **Private cloud:** The cloud resources are used only by one customer. It could be managed by the same company or a third party.
2. **Community cloud:** Same as private cloud, but the customer could be a community or a group of people having the same area of interest.
3. **Public cloud:** The cloud resources are used within the general public.
4. **Hybrid cloud:** The cloud resources could use two or more deployment models. For example, the storage could be in the private cloud, and the computation could take place in the public cloud.

2.2.5 Mobile Cloud Computing

The improvement of cloud computing technology and mobile services have led to the idea of mobile cloud computing (MCC), which aims to move the computational services from mobile users devises to the cloud. This will allow a small mobile device to use the huge capabilities in the cloud. Previous studies mostly defined MCC as

"An infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and MC to not just smartphone users but a much broader range of mobile subscribers" [19].

These devices connect to the cloud with wireless connections, and there will be partitioning and offloading of the computational services [20].

The architecture of MCC consists of three major technologies: mobile computing, the internet and cloud computing [21]. Figure 2.2 [20] shows that a mobile device connects to the mobile network through either a base transceiver, access point or satellite. Thus, mobile devices can connect to the cloud via the internet and use cloud services from a cloud service provider [19].

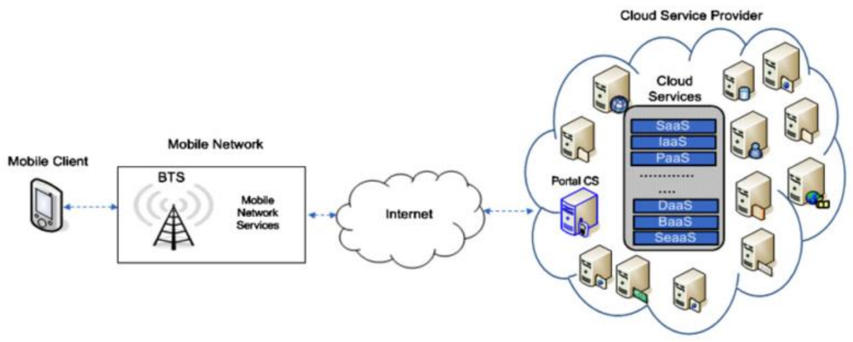


Figure 2. 2: Mobile Cloud Computing Architecture [16].

MCC is generally assumed to play a role in the utilisation of cloud computing resources by allowing mobile users to use them [22]. MCC has contributed to improving mobile computing by providing the following advantages: mobile users have a massive amount of resources, the integration and scalability of mobile applications become more accessible and on-demand services [19]. There are several types of MCC applications. Mobile learning is one example. Educational content can be delivered to students by combining mobile computing and cloud computing. Mobile learning used media such as video, audio, or chat to provide an environment for learning [23]. Another example is mobile gaming, which has increased recently because of the improvement of interactive video gaming [24].

2.2.6 Limitations

Studies have consistently shown that cloud computing does not deal well with latency-sensitive applications such as self-driving cars, health-care applications and video gaming [25]. Moreover, cloud computing might not be considered an efficient computing model for applications that require mobility support and location awareness. Such applications have sharply increased and will continue to increase the cloud’s load [26]. Since the cloud is far from the user, the delay of transferring a huge amount of data, such as video with high resolution, to be processed to the cloud and then back to the edge device is not efficient [27]. Moreover, a challenging critical issue is that cloud computing has full responsibility from the cloud to the end devices [28]. Such responsibility might contribute to the increased use of energy because of transmitting data over multiple hubs from the end devices to the cloud as well as performing all the computations in the cloud [29].

Despite the success of MCC in improving mobile computing, however, the network becomes overheated because the cloud is far away from the mobile user, and this leads to long latency [30]. Thus, some sensitive applications cannot work effectively with the cloud [31]. To conclude, moving a massive amount of data from end devices to the cloud and vice versa is costly in terms of time and energy. Also, it could be infeasible due to the growth of data size and the number of connected devices. Therefore, recent studies have introduced a system model that aims to overcome these challenges, which will be discussed in the next section.

2.3 Edge Computing

This section details the literature of edge computing, its definitions and characteristics. It also provides a comparison between the edge and the cloud, discusses the importance of the edge, presents some of the current edge computing models. Cloud computing, mobile computing and mobile cloud computing are revolutionary technologies, but they require hosting the services only in the cloud, which is maybe impossible for some applications. Thus, a new approach has arisen called edge computing.

2.3.1 Definition

Several papers[32][33][34] state that edge computing is a term that aims to push computational services from the centralized data centre/cloud to the edge of the network to reduce latency and provide real-time interaction as well as supporting the massive growth of connected devices to the internet. OpenEdge Computing defines edge computing as computation provided by small data centres located closer to IoT devices at the edge of the network. Furthermore, provide all the cloud service (i.e. compute and storage) in a virtualized manner [33]. Cloud and edge complement each other and have nearly the same functionality to provide computing services. Yet, there are some differences such as location, support mobility, heterogeneity and scalability to accommodate a vast number of connected devices [28] (table 2.1 summarise the differences).

Table 2. 1: Cloud Computing vs Edge Computing

Features	**Cloud Computing**	**Edge Computing**
Computational Capacity	High	Medium to low
Latency	High	Low
Mobility supported	Limited	Supported
Location of servers	Within the internet	Close to end devices
Number of servers	High	Few
Geographical distribution	Centralized	Decentralized
Suitable for applications require	Intensive computational and delay-tolerant.	Latency-sensitive, mobility and high QoS.

Edge computing has several advantages that will improve the process of distributed systems. First, it reduces the load in the cloud, which in turn will reduce the latency and produce faster response times because it reduces the movement of data from end device to the core of the cloud [27]. Moreover, according to recent studies [35], edge computing could reduce the energy consumption of the cloud up to 40%, which is a significant motivation due to current concerns about energy consumption [36].

Also, edge computing provides a vast amount of resources to IoT devices. Thus, IoT devices become smarter by processing complex tasks in a short time. It is difficult for such devices to handle these tasks by their own because of their limitations, such as computational power [37]. Moreover, regarding the massive increase in the number of devices connected to the internet, edge computing provides scalability to support these devices and deal with their requests closer to them [35]. Not only that, many current applications demand mobile support such as connected vehicles, transport applications and health-care applications [38].

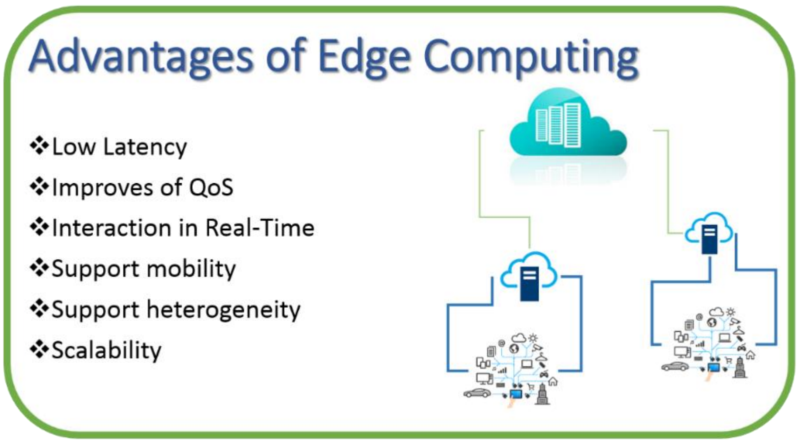


Figure 2. 3: Advantages of Edge Computing

2.3.2 Architecture

Edge computing consists of several edge nodes that are distributed geographically. These edge nodes follow the same concept as cloud computing but in a smaller size, where each node has its computational power, storage, and network. Several studies have called them micro clouds [39][40] or micro data centres [41][42].

In an edge computing environment, there are three main layers (see figure 2.4). The lowest layer contains smart end devices which have limited computational power. Devices in this layer have their functions (e.g. health-care devices, self-driving cars, sensors and smartphones). These devices are connected to the middle layer, which has edge nodes. Edge nodes are close to the end device and provide the required computational resources to the end devices on demand. Edge nodes aim to reduce the latency of IoT applications and dependability to the cloud.

Moreover, edge nodes have limited computational power that may be required to collaborate with either the cloud or other edge nodes. The cloud has enormous computational resources, but it is far away from the end devices, which causes network delays. In an edge computing environment, the cloud manages edge nodes and help if the edge nodes require more computational support or applications that are not supported in the edge node.

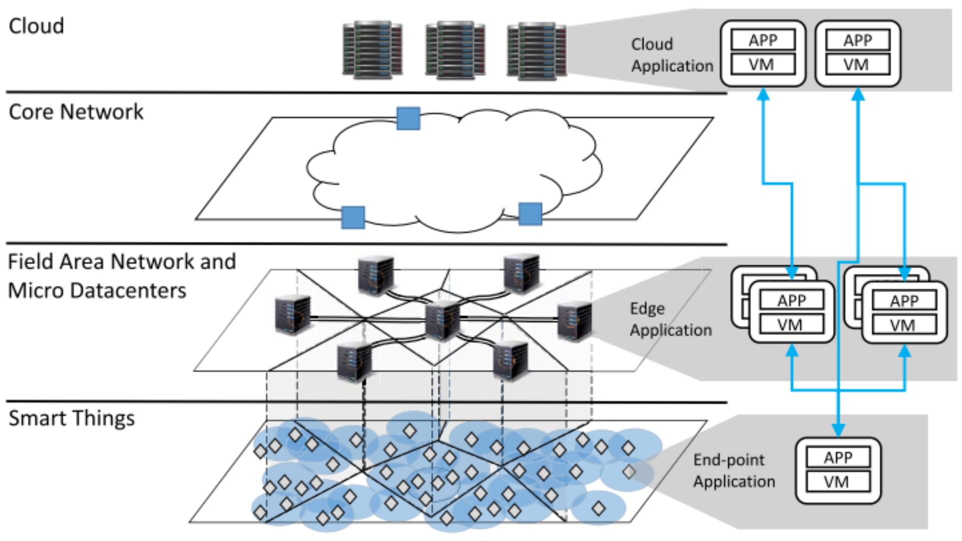


Figure 2. 4: Layers of Edge computing [42].

2.3.3 Related Computing Paradigms

Edge computing is a model that complements the responsibility of the cloud [43]. Therefore, mobile edge computing, fog computing and cloudlet are models that have nearly the same concept [44][45]. Table 2.2 compares these technologies in terms of architecture and suitable applications [46]. The resources of edge computing can be owned by cloud providers or any other vendor such as mobile network providers, university campuses or coffee shops [47]. This section will discuss why there is a need for edge computing and describe these technologies in depth.

2.3.3.1 Cloudlets/Micro-Cloud

Cloudlet, which proposed by Carnegie Mellon University, is another model of Edge computing. Satyanarayanan defines cloudlets as "*a trusted, resource-rich computer or cluster of computers that's well-connected to the Internet and available for use by nearby mobile devices*" [48]. Thus, it is a small cloud that aims to help sensitive applications of users' mobiles, such as gaming, GPS routing, and internet banking [25]. The idea of cloudlets or micro-clouds arose to push computations from centralized cloud systems closer to the end-users' mobile, to avoid the high latency of an offloading approach because these clouds are located far from the end devices [30] [45].

The architecture of cloudlets has three levels (see Figure 2.5). The lowest level is the users' devices that are connected to the cloudlet. The middle level is between users' devices and the cloud. Thus, users are not required to communicate with the cloud directly, which is far away. Cloudlets can communicate with the centralized cloud for configuration and provisioning

[49]. Mobile users can connect to cloudlets through wireless LAN [48]. Compared to the central cloud, a cloudlet is smaller, closer to end-users, and saves power and costs [50]. Thus, it will help mobile applications reduce overall latency and improve Quality of Service (QoS) [51].

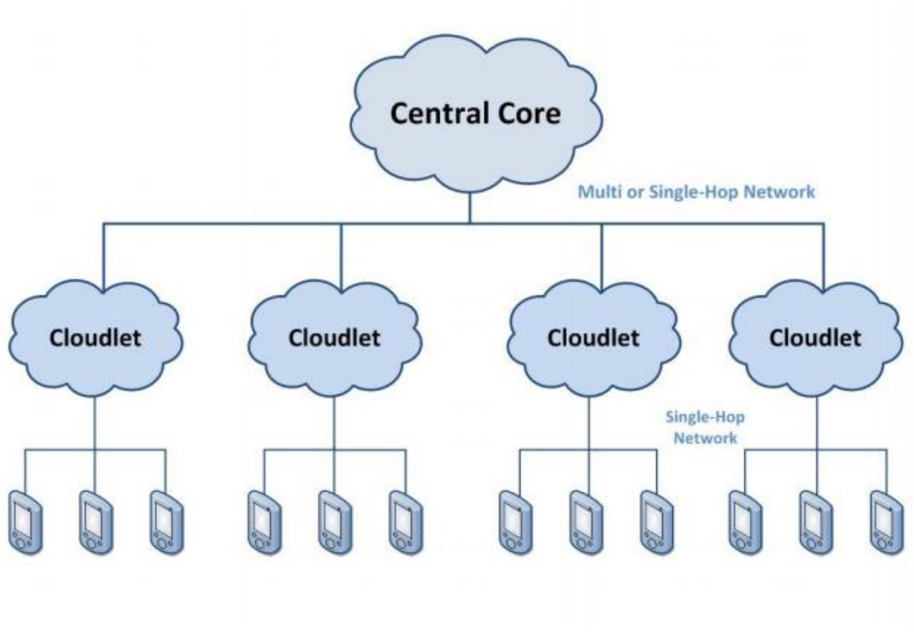


Figure 2. 5: Architecture of Cloudlet [52]

Although the concept of cloudlets is succeeded to support and reduce communication latency, the model is not considered scalable in resource provisioning and services. Moreover, it can only be accessed by Wi-Fi, which causes a limitation to support other devices that are close to a Wi-Fi area but are not covered [53].

2.3.3.2 Fog Computing:

Fog computing is an extended model of cloud computing—“*the cloud close to the ground*”—to serve the edge of the network. It distributes computing resources such as processing units, storage, and networks in the area between the cloud and end devices and has the same techniques in the cloud, such as virtualization and multi-tenancy [54]. Fog computing provides better services to applications and services that do not work effectively with the cloud. These applications have different attributes (e.g. mobility support and real-time interaction), thus require different approaches to work with [55].

Many applications can benefit at least in part from fog computing, including video conferencing, online gaming and AR/VR applications.

The architecture of fog computing consists of four primary levels: data centre cloud, the core of the network, edge node and smart thing IoT [55] (see Figure 2.6). The intermediate layer, the edge node, plays an essential role in supporting the cloud to reduce a load of computing, storing and networking and provides the services to end-users with high QoS. These edge servers are virtualized and can be accessed by connected devices through wired or wireless connections. The edge server connects to the cloud to collaborate with some services [56].

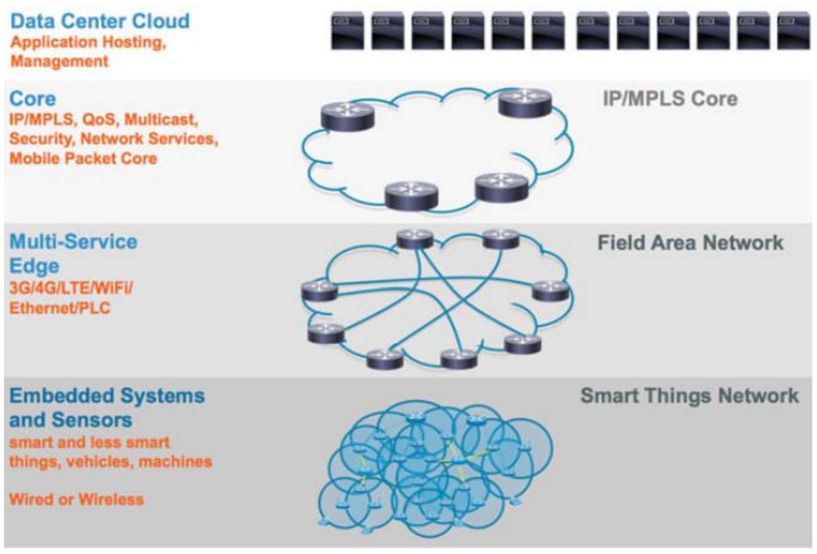

Data Center Cloud
Application Hosting, Management
Core
IP/MPLS, QoS, Multicast, Security, Network Services, Mobile Packet Core
IP/MPLS Core
Multi-Service Edge
3G/4G/LTE/WiFi/ Ethernet/PLC
Field Area Network
Embedded Systems and Sensors
smart and less smart things, vehicles, machines
Wired or Wireless
Smart Things Network

Figure 2. 6: High Level Architecture of Fog Computing [51]

As described earlier, fog computing supports various type of applications, particularly those applications that require real-time analysis and interaction [57]. Thus, the majority of IoT applications are supported by fog computing (e.g. smart home, health care, smart factories, agriculture, etc.) [58].

2.3.4 Mobile Edge Computing:

Mobile Edge Computing and also known as Multi-access edge computing, is defined by European Telecommunications Standards Institute (ETSI) as a technology that

“provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the Radio Access Network (RAN) and in close proximity to mobile subscribers”[59].

As previously stated, the requirements of mobile applications are changing, and new network technologies have appeared, such as 5G. Thus, redesigning the network or the way they provide services is essential [60]. For example, video streaming in the area of smart cities requires a prober network to carry a massive amount of data. Also, the edge server deals with these data near the source [50]. One main difference between mobile edge computing and fog computing is that the former can provide services to connected users without communicating with the cloud in some applications [61].

Previous papers [53][62] list some main characteristics of mobile edge computing. The location of the edge server can be accessed within the range of Radio Access Network (RAN) to provide real-time interaction (see figure 2.7). These edge servers are distributed geographically to support large-scale systems. Thus, mobile edge computing has been recognized to be the base for latency-sensitive applications (e.g. video streaming), also provide the support for IoT mobility and location awareness (e.g. smart vehicle).

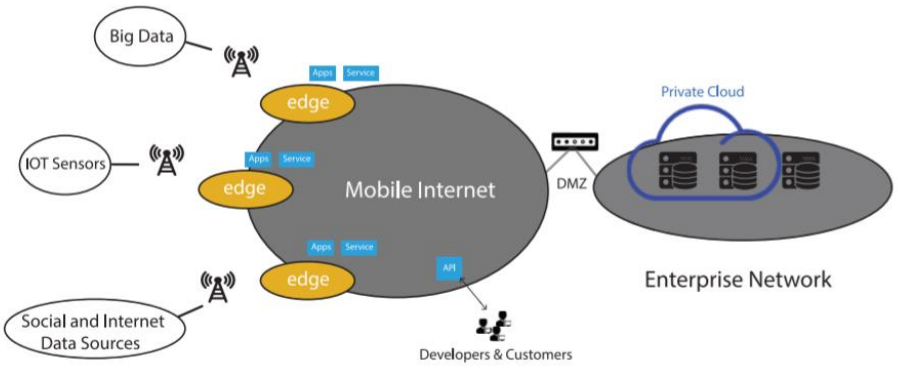


Figure 2. 7: Architecture of mobile edge Computing [63]

Table 2. 2: Comparison of Cloudlets, Fog Computing and Mobile Edge Computing.

	Cloudlets/ Micro-Cloud	**Fog Computing**	**Mobile Edge Computing**
Location	Within the area	Between Edge of Network and the cloud	Radio Network / Base station.
Proximity	Single Hop	Single or several Hops	Single Hop

Devices	Server	Router, Access points, Server	Servers in the base station
Accessibility	Wi-Fi	Wi-Fi, Mobile Network, etc.	Mobile Network
Application	Suitable for mobile applications that require low latency.	IoT	Suitable for applications that require mobility support such as a self-driving car.

Based on the characteristics stated in the previous subsections, Table 2.2 presents a comparison of the three computing paradigms. The site deployment of Cloudlet and MEC can be at the first single hop, for example, Cloudlet can be located indoor within (e.g. a shopping centre, hospital, etc.) and the MEC server can be embedded into the telecom's base station [64]. In contrast, the deployment of Fog computing can be anywhere between the IoT devices and the cloud. The concept of Fog computing is used widely in the applications of smart cities, smart grids, etc.[54]. MEC is used usually in the area of applications that require mobility such as autonomous vehicles and supporting communications of a vehicle to vehicle (V2V) as well as vehicles to Infrastructure (V2I) [65][66]. From the research perspective, all of the above terms have the same concepts, which push the computational service to the end of the network, but in the industrial side, each vendor (e.g. Cisco, Juniper) argues that their devices (e.g. routers and switches) are the perfect platforms to host Edge-Cloud capabilities. On the other hand, telecom companies argue that their base stations and 4G/5G will be hosting the Edge-Cloud capabilities [67].

2.4 Internet of things

There are several definitions and concepts that stand behind IoT. For example, IoT (Internet of Things), IoE (Internet of Everything), and CoT (Cloud of Things) are referring to the same concept. IoT produces a set of new applications for the next wave of the ICT sector. Edge computing has been proposed to deal with the huge change in the area of the distributed system. Recently, the number of devices that are connected to the internet (IoT) has increased massively, and some studies predict that in the upcoming three years, more than 50 billion devices will be connected to the internet [68][69]. This called Internet of things IoT, which is generally can be defined as:
“scenarios where network connectivity and computing capability extends to objects, sensors and everyday items not normally considered computers, allowing these devices to generate, exchange and consume data with minimal human intervention”[70].

2.4.1.1 Characteristics

In terms of characteristics, Figure 2.8 presented the fundamental of IoT characteristics. IoT devices are heterogeneous in terms of hardware, network technologies and platforms. Each IoT device has a unique address and can communicate with others, generate and process data. Also, it can be designed to do any functions as well as integrate with any technology.

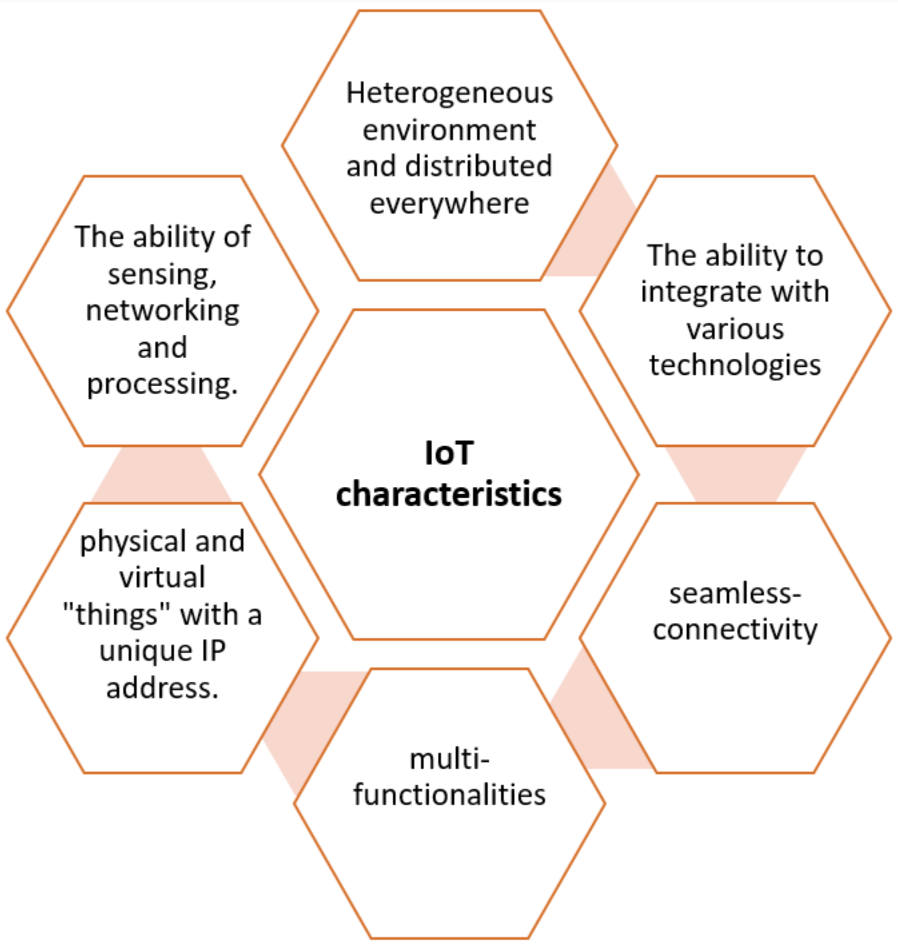

Heterogeneous environment and distributed everywhere
The ability of sensing, networking and processing.
The ability to integrate with various technologies
IoT characteristics
physical and virtual "things" with a unique IP address.
seamless-connectivity
multi-functionalities

Figure 2. 8: Characteristics of Internet of things

2.4.1.2 Enabling technologies

The key enabling technologies of IoT can be classified as follows: embedded system, network technologies, Big Data analytics and Cloud Computing services. First, The embedded system can be defined as a sensor, processor or connectivity antenna in any everyday object. Second, enabling technologies in the network includes communication protocols, network hardware and type of network such as 5G [71]. Third, the developing systems of Big Data analytics (e.g. Hadoop and Spark) [68]. Finally, Cloud Computing and extended models such as Edge computing for computation and storage services [51].

2.5 Scheduling offloading tasks in Edge-Cloud Environments

Computation offloading is not a new paradigm; it is widely used in the area of cloud computing. Offloading transfers computations from the resource-limited mobile device to resource-rich cloud nodes in order to improve the execution performance of mobile applications and the holistic power efficiency. User devices are evenly located at the edge of the network. They could offload computation to Edge and Cloud nodes via WLAN network or 4/5G networks. Broadly, if a single Edge node is insufficient to deal with the surging workloads, other Edge nodes or Cloud nodes are ready for assisting such application.

Basically, it is a solution to support IoT applications by transferring heavy computation tasks to powerful servers in the edge-cloud system. It is a technique used to overcome the limitations of IoT devices in terms of computation power (e.g. CPU, memory, etc.) and insufficient battery. It is one of the most important enabling techniques of IoT because it allows performing a sophisticated computational more than their capacity [72]. The decisions of computational offloading in the context of IoT can be summarised as follows. First, whether the IoT device decides to offload a computational task or not. In this case, several factors could be considered, such as the required computational power and transferred data. Second, if there is a need for offloading, does partial offloading or full offloading. Partial offloading refers to the part of the tasks will process locally at the IoT device and other parts in the Edge-Cloud servers. Factors such as tasks dependency and tasks priority can be considered in this case. Full offloading means the whole application processed remotely in the Edge-Cloud servers [63].

In terms of the objectives of computation offloading in the context of Edge Computing, it can be classified into two categories, objectives that focus on application characteristics and objectives that focus on Edge-Cloud resources. Several studies exist in the literature aim to reduce service latency, minimize energy consumption, maximize total revenue, minimize mandatory cost and maximize resource utilization. In fact, scheduling offloading tasks is a challenging issue in the Edge Computing paradigm since it should consider several trade-offs form application requirements (e.g. reduce latency) and system requirements (e.g. maximize resource utilization). Thus, developing appropriate and efficient resource management which can meet the requirements of both application and system is attracting many researchers in the filed [73][74][75][76].

In the following subsections, some of the studies conducted on task offloading in edge computing to reduce the latency and maximise resource utilization will be discussed.

2.5.1 Task offloading based on Application Characteristics

With the increase of IoT applications, scheduling offloaded tasks that focused on application characteristics is considered significantly important, as highlighted in [77][4][78]. The section below presents the existing literature on task offloading is extensive and focuses mainly on application characteristics: computation, communication and latency-sensitivity.

2.5.1.1 Computation and Communication Demands

There are many ongoing research projects focusing on the tasks computation and communication demands of IoT applications. For example, Wang et al. [79] proposed an online approximation algorithm that main objective to balance the load and minimizing resource utilization to enhance application performance. This work considers computational and communications attributes, however, it does not consider the service latency as well as their solution for homogenous resources. Rodrigues et al. [80], presented a hyper method for minimising service latency and reduce power consumption. This method aims to reduce the communication delay and computational delay by migrating the VM to the unloaded server. The authors investigate the impact of tasks computational and communication demands. They evaluate their approach under realistic conditions by mathematical modelling. However, their method does not consider the application delay constraints as well as the offloading to the cloud. Deng et al. [81], proposed an approximate approach for minimising network latency and power consumption by allocating workload between fog and cloud. However, this approach does not optimise the trade-off between all mentioned objectives (e.g. computational delay and resource utilisation).

Zeng et al. [82] designed a strategy for task offloading that aims to minimize the completion time. In their work, both computation time and transmission time are considered. The authors investigate the impact of other factors such as I/O interrupt requests and storage activities. However, this work does not consider delay-constraints applications and resource heterogeneity. Fan et al. [83] designed an allocation scheme aim to minimise service latency for IoT applications. Their algorithm takes account of both computation delay and communication delay. The authors investigate the impact of the overloaded VM on processing time and evaluate their work with different types of applications. However, the proposed method does not show the effectiveness of the heterogeneity of the VMs in terms of service time does not consider the latency-sensitive application.

2.5.1.2 Latency Sensitivity

In terms of application latency-sensitivity, a number of studies are conducted in order to enhance the overall service time in Edge-Cloud environment. For example, Mahmud et al. [77] proposed a Latency-Aware policy that aims to meet the required deadlines for offloading tasks. Further, it considers the resource utilization at the edge level. This approach considering task dependency as well as the computational and communication requirements. However, resource heterogeneity dose not addressed in their research. Azizi et al. [84] designed a priority-based service placement policy that prioritises tasks with deadlines; thus, nearest deadlines scheduled first. Also, their work considers both computational and communication demands. However, their evaluation dose not addressed when the system has multi IoT devices as well as resource utilisation. Sonmez et al. [85] presented an approach for task offloading that targeting latency-sensitive applications. This approach based on fuzzy logic, which focused on delay as an important factor along with computational and communication demands. However, this approach does not consider resource heterogeneity.

This section has presented the literature of offloading tasks that consider mainly application characteristics and has argued that The next part of this chapter will present the studies that focused on resource utilisation or heterogeneity as the main objectives.

2.5.2 Task offloading Based on Edge-Cloud Resources

2.5.2.1 Resource Utilization

Scheduling offloading tasks based on resource utilisation or resource heterogeneity has received considerable critical attention from many researchers. For example, Nan et al. [86] developed an online optimisation algorithm for offloading tasks that aim to minimise the cost of renting cloud services by utilising resources at the edge based on Lyapunov technique. Further, their algorithm guarantees the edge service availability and ensure to process the task with the required time. However, this algorithm does not consider the impact of computational and communication demands for latency-sensitive applications. Xu et al. [42] proposed a model for resource allocation that aims to maximise resource utilisation and reduce task execution latency. The authors aim to reduce the dependability on the cloud, thus reduce cloud cost. However, this work only considers resource utilisation and does not refer to resource heterogeneity. In addition, application uploading and downloading data are not addressed in this work, which plays a significant role in overall service time. Li and Wang [87] introduced a placement approach that aims to reduce Edge nodes energy consumption and maximise resource utilisation. They evaluated the proposed algorithm through applied numerical analysis on Shanghai Telecom dataset. However, their work does not provide any information regarding the application characteristics (e.g. computation, communication and delay-sensitivity).

2.5.2.2 Resource Heterogeneity

Considering resource heterogeneity in the Edge-Cloud environment for the offloading decision play a critical role to enhance service time performance. A number of studies have investigated the impact of resource heterogeneity on the service time. For example, Scoca et al. [88], proposed a scour-based algorithm for scheduling offloading tasks that considers both computation and communication parameters. Furthermore, their algorithm considers a heterogeneous VMs and sort heavy tasks to be allocated to the more powerful VM. However, their algorithm does not consider server utilisation as key parameters could affect the performance of service time. Roy et al. [89] proposed a strategy for task allocation that allocating different application tasks to an appropriate edge server by considering the resource heterogeneity. This approach aims to reduce the execution latency as well as balance the load between edge nodes. However, this approach does not consider task communication time. Taneja et al. [90] proposed a resource-aware placement for IoT offloading tasks. Their approach ranks the resources at the edge with their capabilities and then assign tasks to the suitable server based on the task's requirements (e.g. CPU, Ram and bandwidth). However, this method focused on improving application service time performance, but without explicitly considering application latency-sensitivity.

2.5.3 Overall Discussion

The effective mechanisms for scheduling offloading tasks can contribute to minimizing the overall service time of IoT latency-sensitive applications and maximize resource utilization in the Edge-Cloud environment. With the dynamicity of IoT workload demands, Edge-Cloud service providers should strike a balance between utilising Edge-Cloud resources and satisfying QoS objectives of IoT applications. Consequently, efficient resource management mechanisms can be beneficial to enhance both resource utilisation and supporting the latency-sensitive application requirements in term of service time.

The above section has reviewed the existing related work on offloading tasks that are focusing mainly on application parameters such as computation demands, communication demands and latency-sensitivity in Edge-Cloud environments. As discussed earlier, the related works in [80][82][83] consider application parameters in order to minimise the service time. However, these works lack to consider the impact of resource parameters such as server utilisation and VMs heterogeneity.

On the other hand, section 2.6.2, presented the work in [79][88][90] which considered the resource utilisation and resource heterogeneity as key objectives in the process of scheduling offloading tasks in the Edge-Cloud environment. Although, some related works such as [81][87][89] have considered application requirements (i.e. computation or communication) but, without explicitly considering the latency-sensitivity of IoT applications.

Hence, there is still a need for efficient resource management that takes into account characteristics of offloading IoT applications' tasks (competition, communication and latency), as well as resource parameters such as resource utilization and resource heterogeneity in order to meet the required service time for the application and utilising Edge-Cloud resources.

The following Table 2.3 provides a comparison summary of the similar related work on scheduling offloading tasks that consider both application characteristics and resource parameters in Edge-Cloud environment.

Table 2. 3: Comparison of individual papers addressing computation offloading decisions

Criteria / by	**Objective**	**Application characteristics considerations**			**Edge-Cloud resources considerations**			**Evaluation method**
		Compute	**network**	**delay**	**Resource utilization**	**Resource type**	**# of devices**	
[79]	Minimizing resource utilization	Considered	Considered	Not considered	Considered	Homogeneous	-	Simulation
[80]	Minimizing service latency	Considered	Considered	Not considered	Considered	Homogeneous	Single	Mathematical
[81]	Minimizing network latency	Not considered	Considered	Not considered	Considered	Homogeneous	-	Simulation
[88]	Minimizing service latency	Considered	Considered	Not considered	Not considered	Heterogenous	Multi	Simulation
[86]	Minimizing cost	Not considered	Not considered	Considered	Considered	Homogeneous	Multi	Simulation
[89]	Minimizing execution time	Considered	Not considered	Not considered	Not considered	Heterogenous	Single	Direct experiment
[82]	Minimizing completion time	Considered	Considered	Not considered	Not considered	Homogeneous	Multi	Simulation
[77]	Minimizing service latency	Considered	Considered	Considered	Considered	Homogeneous	Multi	Simulation
[84]	Minimizing service latency	Considered	Considered	Considered	Not considered	Homogeneous	Single	Simulation

[85]	Minimizing service latency	Considered	Considered	Considered	considered	Homogeneous	Multi	Simulation
[90]	Minimizing service Time Maximise resource utilization	Considered	Considered	Not considered	Considered	Heterogenous	Multi	Simulation
[83]	Minimizing service Time	Considered	Considered	Not considered	Not considered	Homogeneous	Multi	Simulation
[42]	Maximise resource utilization	Considered	Not considered	Considered	Considered	Homogeneous	-	Simulation
[87]	Maximise Energy Maximise resource utilization	Not considered	Not considered	Not considered	Considered	Homogeneous	Multi	Data-Driven Analysis

2.6 Open Challenges

As far as offloading tasks is concerned, several open challenges require numerous efforts to address. This section will present open challenges of offloading tasks in Edge-Cloud environments.

- Task dependency: There are lacks of the studies addressing the problem of offloading tasks because they do not consider the dependency of the tasks. To be more precise, allocating tasks that are dependent on the results of other tasks to different resources in the edge-cloud could lead to poor QoS for IoT applications. It requires to study the application components and how there interact with each other. Considering this factor could lead to enhance both the overall system performance and satisfy application QoS.

- Applications require a high degree of mobility: Offloading tasks of applications that require mobility support such as a self-driving car, crewless aircraft vehicles, and mobile devices, is an open challenge. For example, processing tasks of application users while moving from covered area to other covered are could lead to high network latency or process failure [4]. Although there are several researchers tackling this issue, however, it is still challenging.
- Workload prediction: IoT tasks dynamically change; thus, each task's procedure may have different execution time. Also, IoT devices are mobile, the number of devices may increase in some area; thus, the workload will be increased for the connected edge node. Hence, the amount of IoT workload will change dynamically over the edge-cloud system, which could lead to service performance degradation. Therefore, there is a need for workload predication, which can help to satisfy application QoS and maintain the performance of the Edge-Cloud system.

This thesis will be focused on the impact of different computational and communication demands as well as resource utilisation and heterogeneity at the edge level. The process of scheduling offloading tasks that consider the previous parameters is an open challenge and attracting many researchers in the field.

2.7 EdgeCloudSim

Quantifying and analysing the performance of applications that running in real Edge-Cloud environment are challenging because of three main issues: (i) IoT applications have varying needs in terms of computational and communications, (ii) it widely geographically distributed as well as demanding mobility support and (iii) resources heterogeneity across (Edge and Cloud). Consequently, it would be costly and time-consuming to conduct a number of experiments across a large-scale Edge-Cloud environment.

An alternative method which more feasible and has been widely used by researchers in this area is the simulation. Simulation tools make it possible to investigate and evaluate research's hypotheses or provide novel solutions in a controlled platform that allows to conduct experiments and get results in a timely manner. It could help to reduce the time and cost of experiments repeatability and collecting results which pave the ways before deploying the solutions in a real environment. Furthermore, it provides an evaluation on the heterogeneous of IoT applications' workloads and Edge-Cloud resources which be rare by direct experiments due to cost, time and technical expertise. Several studies [91][14] have stated that edge-cloud simulation tools are recommended to have the following features: (1) support cloud services, such as the scheduling and provisioning of VM; (2) support network functionalities, such as a modelling network in each layer (LAN, MAN, and WAN); (3) support applications services (i.e. computational and communication demands, delay sensitivity, etc.); and (4) support edge services, such as the mobility of IoT devices and managing edge nodes.

In general, there are several issues should be considered in order to select the appropriate simulation tools and validate the proposed approach and its results. This will help to overcome and mitigate the general limitations of the simulation-based method. These issues are the following:

The selection of the appropriate simulator tool: According to [], the first step is to search for the simulator tools that have the necessary functionalities to deliver the research aims and objectives. Other criteria, such as (the required HW to run the simulation, the efficiency of generated logs files, the cost of simulation and assistance availability), could also be used to select the simulation tool [15]. In this research, we reviewed a collection of the existing tools (e.g. IfogSim, IoTSim, and FogNetSim++), comparing their main focuses. EdgeCloudSim was the most appropriate for the purpose of this research. More details are in section 2.7.

The selection of simulation parameters: In a simulation-based method, the generated result depends on the selected parameters, and therefore, any wrong parameters could lead to incorrect results. In this research, we selected the parameters based on related research in both application demands (e.g. computational and communications) and edge-cloud resources (e.g. VM configuration). Additionally, a sensitivity analysis was conducted to investigate the impact of the application parameters.

The validation of the generated results: In order to avoid any anomalies from the simulation results, each simulation was run five times, and a statistical analysis was conducted to take the mean values. Then, some of the results obtained were compared with the general findings from other research. For example, some of the generated results were consistent with other researchers' results from direct experiments.

There are a number of simulation tools that suggested in the literature such as IfogSim [92], EdgeCloudSim[11], IoTSim and FogNetSim++[93]. Each one of these tools has a special focus and characteristics.

- **IfogSim:** is a java-based tool that relies on a well-known cloud simulator CloudSim[94] and provides a multi-layered architecture from IoT to the Cloud. However, the network load has been neglected and does not support the mobility of IoT devices.
- **EdgeCloudSim** also based on CloudSim and it covers mostly all the aspect of the edge computing environment, which include resources in both computation and communication level. Additionally, supporting the mobility of IoT devices[95].
- **IoTSim:** focuses on the processing of Big Data for IoT applications. It is an extension of CloudSim with the following extra layers: Storage Layer, Big Data Processing Layer and Application Layer. It also modelled the MapReduce approach for IoT applications.
- **FogNetSim**++: is relies on OmNeT++[96] simulator. The main focus on network modelling of fog network and provides the ability to implement scheduling algorithms of fog networks and hand-over of IoT device. Moreover, supporting mobility.

Comparing with the above simulators, EdgeCloudSim could be appropriate for this research more than the others for the following reasons:

- **Service time**: The main purpose of this research is to investigate and evaluate the performance of IoT application in edge computing environment through the overall service time, which consists of processing delay and network latency. In this research, service time refers to the time for each task that will be handled in the edge computing environment from sending the request to receive the result. EdgeClouSim is different from other simulators such as iFogSim in this context because iFogSim is ignoring the network load by assuming a fixed delay for any network link, which it could not mimic the actual environment.
- **Application characteristics**: In the perspective of Edge-cloud provider, IoT applications have three main features; it is distributed over large geographic areas, mobility and scalability. Scalability refers to the number of IoT in a specific area could increase or decrease for any reasons.
- **Architecture:** Edge Computing has different architectures such as; standalone edge node, edge-cloud which mean the IoT tasks will be offloaded to edge or cloud servers and several edge nodes running with each other in coordination with the cloud. EdgeCloudSim support simulating these architectures[95].

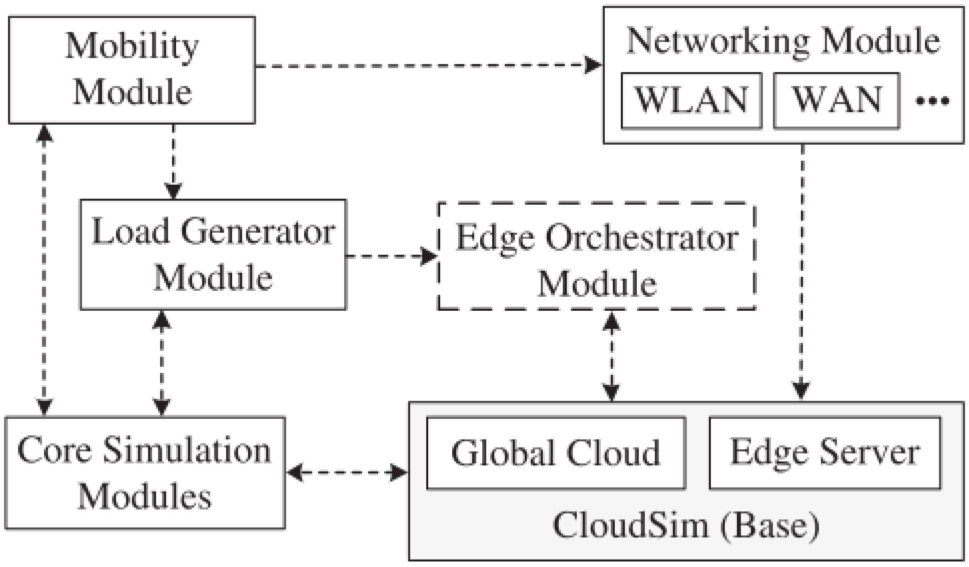


Figure 2. 9: main components of EdgeCloudSim [11]

Additionally, EdgeCloudSim considering the main three aspects of the edge computing environment; Firstly, Computational modelling which covers; Datacenter model, VM provisioning and task execution. Secondly, Network modelling which covers; link properties, delay model data transfer size and network capacity. Thirdly, Edge Specific Modelling which considers; Edge system design, mobility, offloading decision and Edge Orchestration. It also consists of the following five main components [11] as shown in figure 2.9 and for further information:

1- **Core Simulation module:** is responsible for loading and running edge computing scenarios based on configuration files which consist of application characteristics, datacentre specifications and simulation settings. Additionally, providing logs of implemented scenarios in a comma-separated value (CSV) format.

2- **Network module**: is responsible for handling the transmission delay for transferring data in WLAN, MAN and WAN. This considers both uploading data from IoT devices and downloading data to IoT devices.

3- **Edge orchestrator module:** is act as a decision-maker for managing computational resources on the edge layer. For example, terminating the edge VM and decide to offload a task to edge servers or the cloud.

4- **Mobility module:** in charge of updating the locations of IoT devices during the simulation experiment based on the mobility model. This allows IoT devices to move from edge area to other edge areas.

5- **Load Generator module:** is responsible for generating IoT tasks based on Application configurations. For example, specify the amount of data for upload/download for each task as well as the amount of required computational power.

2.8 Summary

This chapter has presented an overview of the fundamental concepts of the subject of scheduling offloading tasks in the edge-cloud system. It started with the core concepts of cloud computing with more details on its definition, architecture, deployment models and the idea of mobile cloud computing as an extended model for cloud computing. Then the core concepts of the transformation to the edge computing and its models are introduced. These presented the idea of edge computing and explain the different terms, such as fog computing, mobile edge computing, etc. This is followed by positioning the work in the related literature, focusing on the scheduling offloading tasks issues and resource management in Edge-Cloud system. A reviewing with related works that focus on application characteristics is presented. Finally, research open challenges and simulation tools are presented.

Chapter 3. Investigating and Modelling Edge-Cloud Environments

3.1 Overview

In this chapter, Section 3.2 presents the overview of Edge-Cloud system architecture that supports scheduling offloading tasks of IoT application, followed by the explanation of the required components and their interactions within the system architecture. Section 3.3 presents offloading latency models that consider computational and communication as key parameters with respect to offloading to the local edge node, other edge nodes or the Cloud. This chapter concludes by discussing early experiments conducted on EdgeCloudSim to investigate the latency models of each offloading scheme as presented in Sections 3.4 and 3.5.

3.2 Modelling Edge-Cloud Environments

3.2.1 System Overview

As illustrated in figure 3.1, the Edge-Cloud system from bottom to the top consists of three layers: IoT devices, multiple edge computing nodes and the cloud. The IoT level is composed of a group of connected devices (e.g. smartphones, self-driving cars, smart CCTV, etc.); these devices have different applications where each application has several tasks (e.g. smart

CCTV [97] application consists of movement dedication, face recognition etc.). These services can be deployed and executed in different computing resources (connected edge node, other edge nodes or cloud), where the infrastructure manager and service providers decide where to run these services.

In our system, at edge level, each edge computing node is a micro datacentre with a virtualised environment. Moreover, it has been placed close to the connected IoT devices at the base station or Wi-Fi access point. These edge nodes have been distributed geographically and could be owned by the same cloud provider or other brokers [98]. It has limited computational resources compared to the resources in the Cloud. Each edge node has a node manager that can manage computational resources and application services that run on. All the edge nodes have connected to the edge controller.

The offloading tasks can be achieved when the IoT devices decide to process the task remotely in Edge-Cloud environments. Applications running on IoT devices can send their offloadable tasks that can be processed by the edge-cloud system through their associated edge node. We assume that each IoT application is deployed in a VM in the edge node and the cloud. IoT devices offload tasks which belong to a predefined set of applications, these tasks are varied in term of the computational requirement (task length) and communication demand (amount of transferred data). It is assumed that tasks are already offloaded from the IoT devices, and each task is independent; thus, the dependency between the tasks is not addressed in this study. The locations of IoT devices are important for the service time performance because it is assumed that each location is covered by a dedicated wireless access point with edge node and the IoT devices connect to the related WLAN when they move to the covered location.

The associated edge can process IoT tasks and also can be processed collaboratively with other edge nodes or the cloud, based on Edge Orchestrator decisions. For example, if an IoT application is located in an edge node faraway from its connected edge, its data traffic has to be routed to it via a longer path in the edge-cloud system. In the Cloud level, a massive amount of resources that enable IoT applications' tasks to be processed and stored.

The proposed architecture is just a possible implementation of other architectures in the literature such as [54][42][69], an example of these architectures represented in Figure 2.4. The main difference in the proposed architecture is the introduced layer between the edge nodes and the cloud. This layer responsible for managing and assign offloading tasks to the edge nodes —more details in the following subsections.

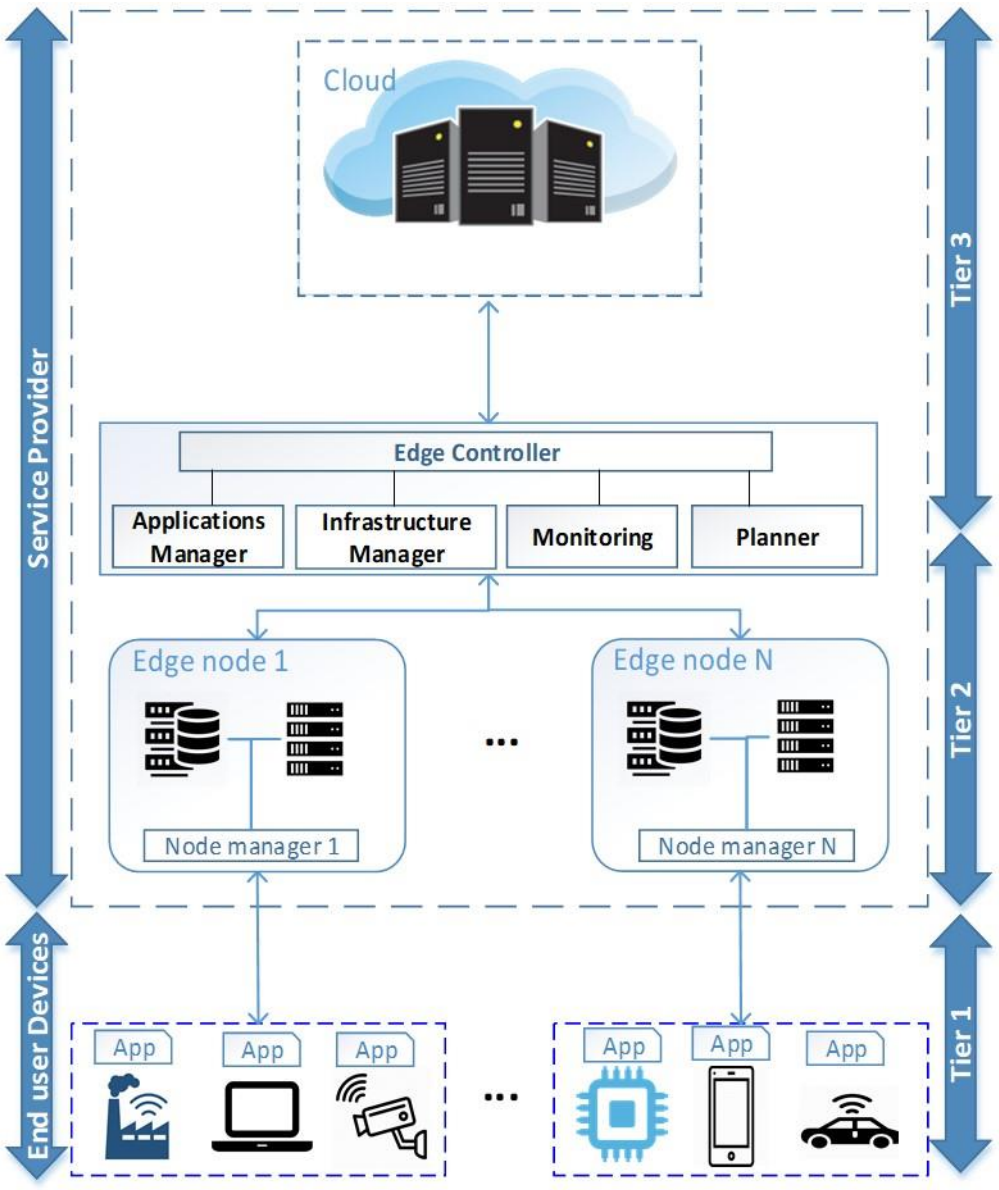


Figure3. 1: An overview of Edge-Cloud system

3.2.1.1 Edge Controller:

Our edge Controller designed similar to [75][99][100], some studies called edge orchestrator, which is a centralized component that responsible for planning, deploying and managing application services in the edge-cloud system. EC communicate with to other components in the architecture to know the status of resources in the system (e.g. available and used), the number of

IoT devices, their applications' tasks and where IoT tasks have been allocated (e.g. edge or cloud). EC consists of the following components: Application manager, Infrastructure manager, Monitoring and Planner. The location of Edge Controller can be deployed in any layer between edge and Cloud. For example, in [101], EC act as an independent entity in the edge layer that mange all the edge nodes in its control. It is also responsible for scheduling the offloading tasks in order to satisfy applications' users and Edge-Cloud System requirements. The EC is synchronising its data with the centralised Cloud because if there is any failure, other edge nodes can take EC responsibility from the Cloud [102][103].

3.2.1.2 Application Manager:

The application manager is responsible for managing applications running in the edge-cloud system. This includes requirements of applications tasks, such as the amount of data to be transferred, the amount of computational requirement (e.g. required CPU) and the latency constraints. Moreover, the number of application users for each edge node.

3.2.1.3 Infrastructure Manager:

The role of the infrastructure manager is to be in charge of the physical resources in the entire edge-cloud system. For instance, processors, networking and the connected IoT devices for all edge nodes. As mentioned above, edge-cloud is a virtualized environment; thus, this component responsible for virtual machines as well. In this research, this component provides the EC with the utilization level of VM.

3.2.1.4 Monitoring:

The main responsibility of this component is to monitoring application tasks (e.g. computational delay and communication delay) and computational resources (e.g. CPU utilization) usage during the execution of applications' tasks in the edge-cloud system. Furthermore, detecting the tasks failures due to network issues or shortage of computational resources.

3.2.1.5 Planner:

The main role of this component is to propose the scheduling policy of the offloading tasks in the edge-cloud system and the location where they will be placed (e.g. local edge, other edges or the cloud). In this research, the proposed approach for offloading tasks in Chapter 5 will work on this component and will pass its results to EC for execution.

3.3 Latency-Sensitive Applications

Applications that have high sensitivity of any delays accrue in communication or computation during the interaction with the Edge-Cloud system. For instance, IoT device sends data to the point that processing is complete at the edge node or the cloud in the back end of the network, and the subsequent communications are produced by the network in response to receive the results. There are many examples of latency-sensitive applications, and the acceptable service time varies depending on the application type which affected by the amount of transferred data and the required computation volume [104]. For example, self-driving cars consist of several services, in [105] classified these services in categories based on their latency-sensitivity, quality constraints and workload profile (required communication and computation). First, critical applications, which must be processed in the car's computational resources, for instance, autonomous driving and road safety applications. Second, high-priority applications, which can be offloaded but with minimum latency, such as image aided navigation, parking navigation system and traffic control. Third, low-priority applications, which can be offloaded and not vital as high-priority applications, e.g. Infotainment, multimedia, and speech processing. Table 3.1 presents more examples of latency-sensitive applications in different technology sectors [104].

Table 3. 1: Latency-Sensitive Applications

Industry	**Applications**
Industrial automation	Industrial Control
	Robot Control
	Process Control
Healthcare Industry	Remote Diagnosis
	Emergency Response
	Remote Surgery
Entertainment Industry	Immersive Entertainment
	Online Gaming
Transport Industry	Driver Assistance Applications
	Autonomous Driving
	Traffic Management
Manufacturing Industry	Motion Control

	Remote Control
	AR and VR Applications

3.4 Edge-Cloud Latency Models

Investigating and modelling the various offloading decisions for IoT tasks can increase the quality of service, which has attracted the attention of many researchers in the field. With the increasing of IoT devices, the amount of produced data, the need for an autonomous system that requires a real-time interaction as well as the lake of support from the central cloud due to network issues, service time has been considered as one of the most important factors to be handled in edge computing [106][81][107].

One of the main characteristics of Edge computing is to reduce the latency level. And it has been proved through literature that using Edge computing will enhance applications performance in term of overall service time comparing to traditional cloud system [108][109][110]. However, different offloading decisions within the edge-cloud system can lead to various service time due to the computational resources and communications types. The current real-world applications measure the latency between the telecommunication service provider and the cloud services[111]. Also, a few existing works compare the latency between offloading to the edge or the cloud. Yet, what about the latency between multiple edge nodes that work collectively to process the offloading tasks. Consequently, investigating the latency of Edge-Cloud system is an essential step towards developing an effective scheduling policy due to the following reasons. First, task allocation in Edge-Cloud system is not only two choices, e.g. either at IoT device or in the cloud, but could be on any edge nodes. Moreover, edge nodes connected in a loosely-coupled way on heterogeneous wireless networks (i.e. WLAN, MAN and WAN), making the process of resource management and the offloading decision more sophisticated. Second, given that task processing is allocated among multiple edge nodes working collectively and the cloud, it is challenging to make an optimal offloading decision.

Therefore, we introduce the latency model to investigate the delay of each offloading scenarios. This section will be exploring the effect of computational and communication for each offloading scenarios. These are: (1) offloading to the local edge, (2) offloading to the local edge with the cloud and (3) offloading to the local edge, other available edge nodes and the cloud. The list of parameters and their notations is shown in Table 3.2.

Table 3. 2: Summary of Notations

Symbol	Meaning
t_{te_up}	Transmission Time between the IoT to the Edge node for uploading
t_{te_down}	Transmission Time between the IoT to the Edge node for Downloading
t_{ce}	Computation time in the edge node
t_{tc_up}	Transmission Time between the Edge node to the Cloud for uploading
t_{tc_down}	Transmission Time between the Edge node to the Cloud for Downloading

t_{cc}	Computation time in the Cloud
t_{teo_up}	Transmission Time between the Edge node to other nearby edge nodes for uploading
t_{teo_down}	Transmission Time between the Edge node to other nearby edge nodes for Downloading
t_{ceo}	Computation time in the other nearby edge node

3.4.1 Latency to Local Edge

This is known as One-Level offloading system which is basically offloading to “Cloudlet” or “Local Edge”. It aims to provide a micro-data centre that supports IoT devices within a specific area such as a coffee shop, mall centre, Airport, etc. [112][113]. Thus, IoT devices can offload their tasks to be processed. This offloading scheme provides ultra-low latency due to the avoidance of network backhaul delays.

As shown in Figure 3.2, IoT devices send their offloading tasks through the wireless network, and then the tasks will be processed by the edge node and finally send the results to IoT devices. The end-to-end service time composed of two delays, network delay and computational delay. The network delay consists of the time of sending the data to edge and the time to receive the output from the edge to the IoT device. The computation time is the time from arriving the task to the edge node until the processing has completed. Therefore, the end-to-end service time latency is the sum of communication delay and computational delay [80], which can be calculated as follows:

$$L_{Local\_edge} = t_{te\_up} + t_{ce} + t_{te\_down}$$

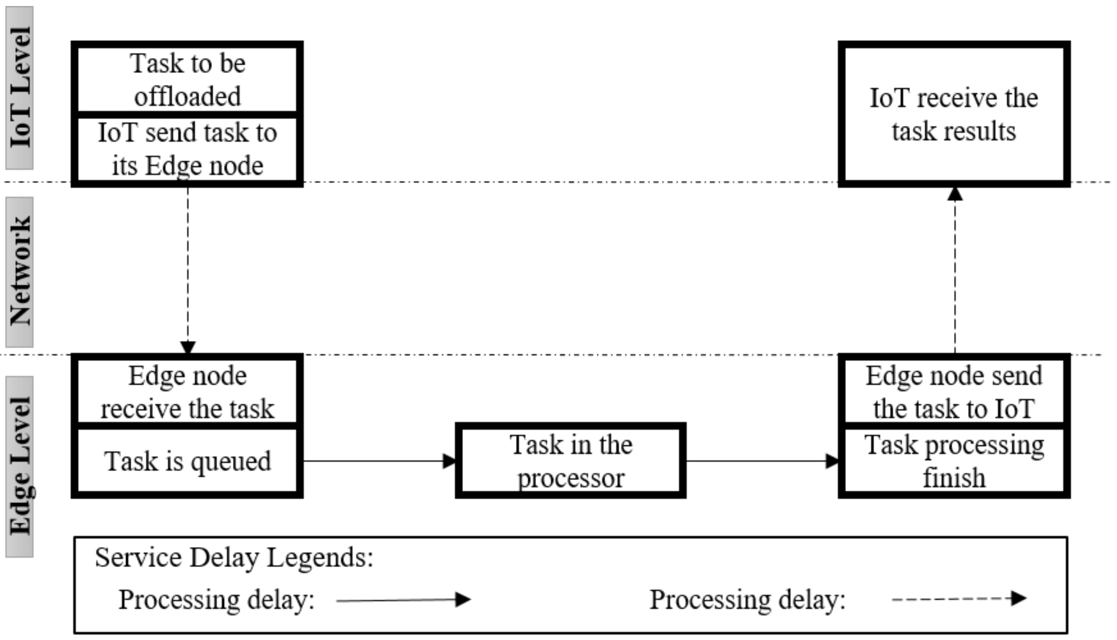


Figure3. 2: Latency to local edge

3.4.2 Latency to Local Edge with the Cloud

In this offloading scheme, rather than relying on only one edge node, The IoT tasks can be processed collaboratively between the connected edge node and the cloud servers. This will combine the benefits of both Cloud and Edge computing, where the cloud has a massive amount of computation resources, and the edge have lower communication time [114]. In this scheme, the edge can do part of the processing such as pre-processing, and the rest of the tasks will be processed in the cloud.

As illustrated in Figure 3.3, IoT sends the computation tasks to the connected edge and then part of these tasks forwarded to the Cloud. Once the cloud finishes the computation, it will send the result to the edge, and the edge will send it to the IoT devices. This scheme consists of Communication time (e.g.

time between the IoT device to the edge node and the time between edge nodes to the cloud) and computation time (e.g. processing time in the edge and processing time in the cloud). Thus, the end-to-end service time can be calculated as follows:

$$L_{L\_C} = t_{te\_up} + t_{ce} + t_{tc\_up} + t_{cc} + t_{tc\_down} + t_{te\_down}$$

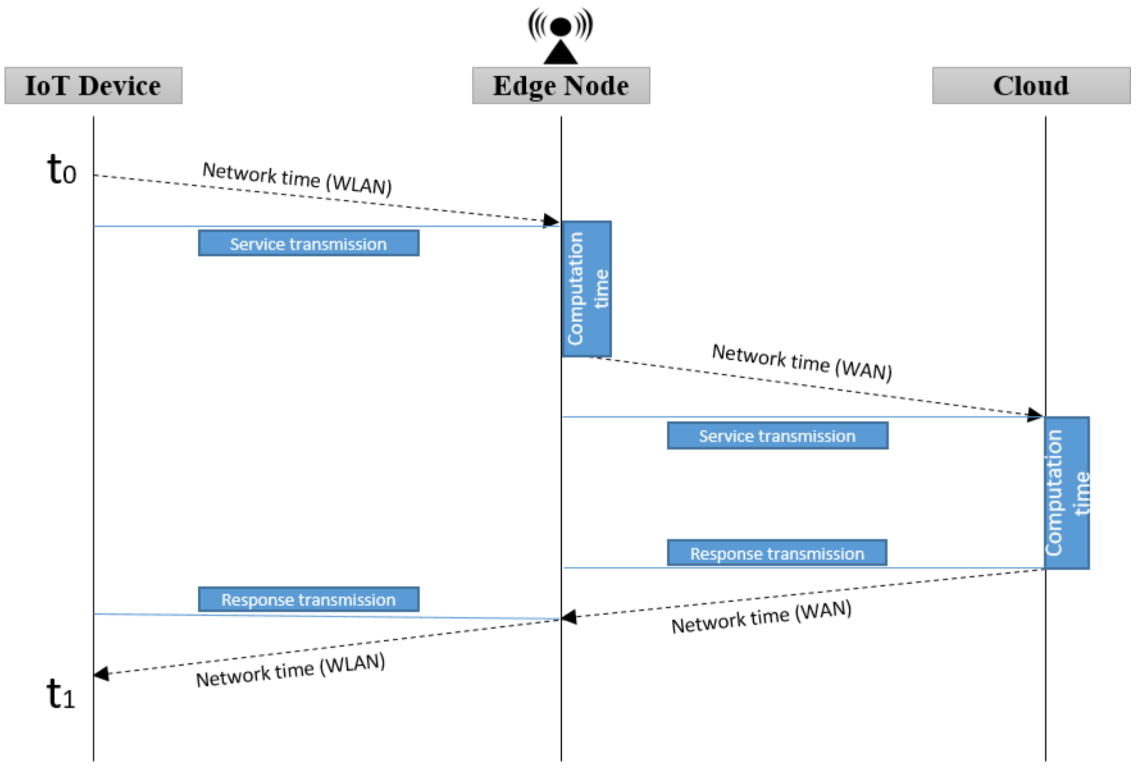


Figure3. 3: Latency to Local Edge with the Cloud

3.4.3 Latency to multiple edge nodes with the Cloud

This is known as a three-level offloading scheme [115] that aims to utilise more resources at the edge layer and support the IoT devices in order to reduce the overall service time. It adds another level by considering other available computation resources in the edge layer. Basically, it distributes IoT tasks over three levels; connected edge, another available edge nodes and the cloud. The edge orchestrator controllers all edge servers by Wireless Local Area Network (WLAN) or Metropolitan Area Network (MAN) which have low latency compared to Wild Area Network (WAN).

As illustrated in figure 3.4, IoT sends the computation tasks to the connected edge and then part of these tasks transferred to other available resources in the edge level through the Edge orchestrator and the rest to the cloud. This will help to decrease the dependency of cloud processing as well as increase the utilisation of computing resources at the edge [110]. This scheme consists of Communication time (e.g. time between the IoT device to the edge node, the time between edge node to other collaborative edge node and the time between edge nodes to the cloud) and computation time (e.g. processing time in the edge, processing time in other collaborative edge node and processing time in the cloud). Thus, the end-to-end service time can be calculated as follows:

$$L_{three-off} = [t_{te_{up}} + t_{ce} + t_{teo_{up}} + t_{ceo} + t_{tc_{up}} + t_{cc} + t_{tc_{down}} + t_{teo_{down}} + t_{te_{down}}]$$

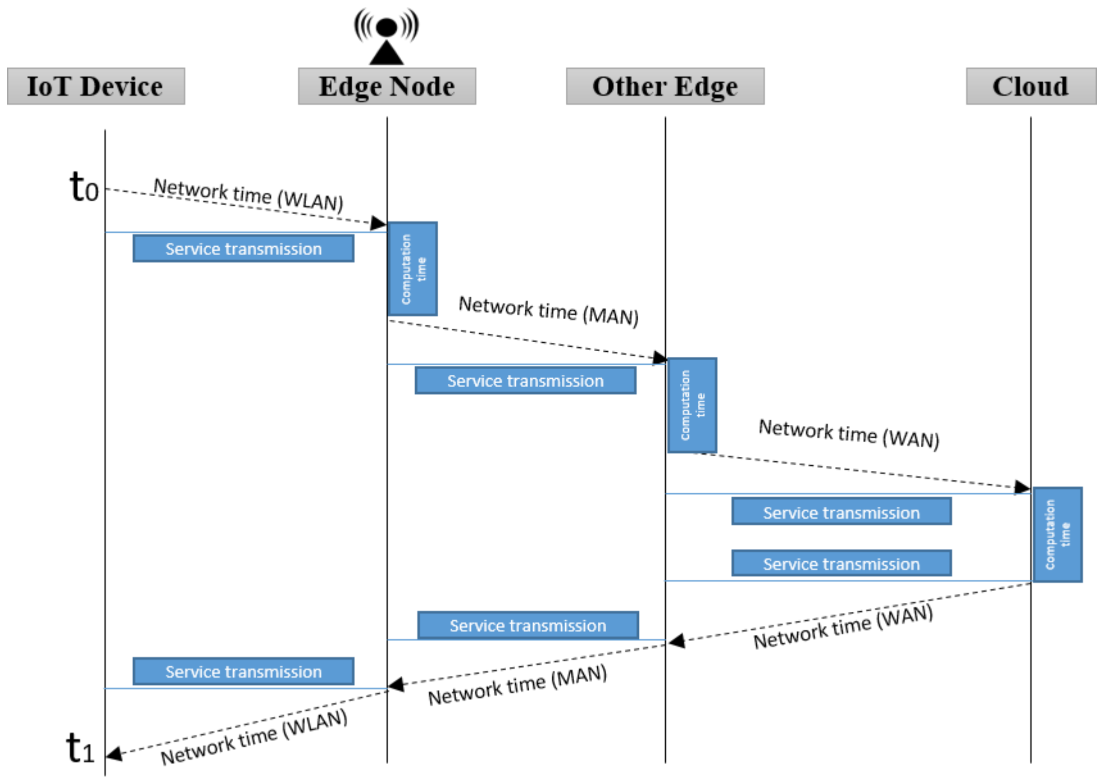


Figure3. 4: Latency to multiple edge nodes with the Cloud

3.5 Early Experiments

In order to obtain an early investigation of the different offloading scenarios, and its influence on overall service time, a number of simulation experiments have been conducted on EdgeCloudSim. EdgeCloudSim provides sufficient models to represent some specific situations. For example, the service time model is designed to represent the several kinds of delay taking place in the WLAN, MAN, and WAN, mobile devices and even the delay of processing in the CPUs of VMs. Thus, experiments of this chapter are practically finished within this simulation to investigate and evaluate the performance of IoT workloads over the three different offloading scenarios. All the experiments are repeated five times, and the statistical analysis is conducted to consider the mean values of the results to avoid any anomalies from the simulation results. We assume that we have three edge nodes connected to the cloud. Each edge node has two servers, and each of them has four VMs. The number of edge nodes does not matter in the context of this research as long it more than two, because one of our aims to investigate the latency between two edge nodes. The cloud contains an unlimited number of computational resources. We got inspiration from other related works such as [80][85] to design the experiments and its parameters (e.g. number of IoT devices, Edge nodes and the amount of transferred data for each offloading tasks). Table 3.3 represents the key parameters of the simulation environment. The warm-up period is used to allow the system to evolve to a condition more representative of steady-state before getting the simulation output. Number of iterations are used to avoid any anomalies from the simulation results.

Table 3. 3: key parameters of the simulation environment

Key parameters	**Values**
Simulation Time	30 minutes
Warm-up Period	3 minutes
Number of Iterations	5
Number of IoT devices	100-1000
Number of Edge Nodes	3
Number of VM per edge server	8
Number of VM in the Cloud	∞

Average Data Size for Upload/Download (KB)	500/500

3.6 Results and Main Findings

The conducted experiments show the results of three different offloading scenarios, offloading to Local Edge (i.e. cloudlet), offloading to Local Edge with the Cloud and offloading to multiple edge nodes with the cloud. The aim of these experiments is to investigate and evaluate the processing delays, network delays and end-to-end service delays of the three offloading scenarios. This will increase our understanding of the offloading decision in the Edge-Cloud system in order to design Edge-Cloud resource management.

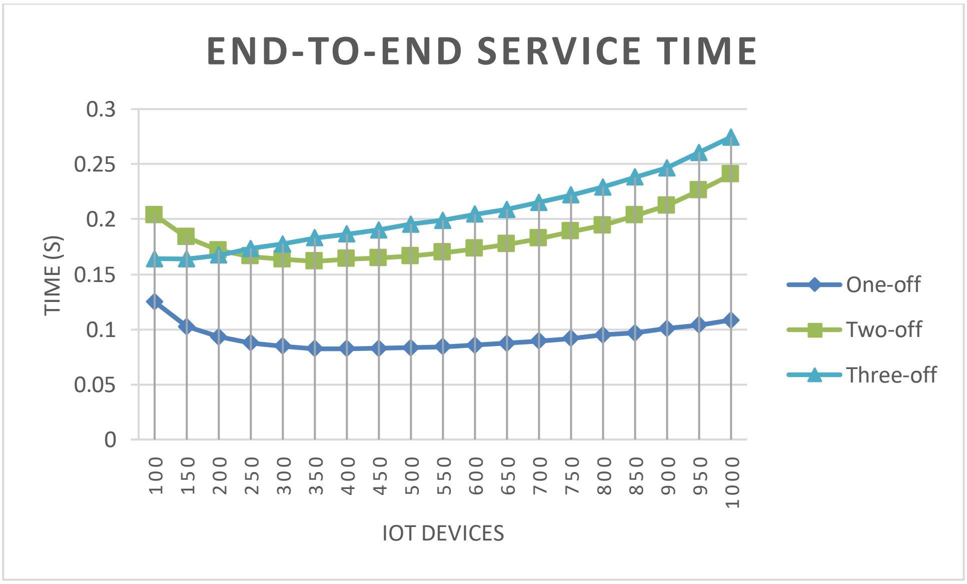


Figure3. 5: End-to-end service time for three offloading scenarios

Figure 3.5 presented the overall service time of the three offloading scenarios. Offloading to one-level is has the lowest service time. This result is consistent with work in [116][80], their explanation of this result because of the avoidance of major latency between the end device and the Cloud. Two-offloading levels have lower service time performance than the three-offloading. This shows the overall service time will never be truly minimised unless the network time is considered in the offloading process. However, these results may be somewhat limited by the number of IoT devices and the system load.

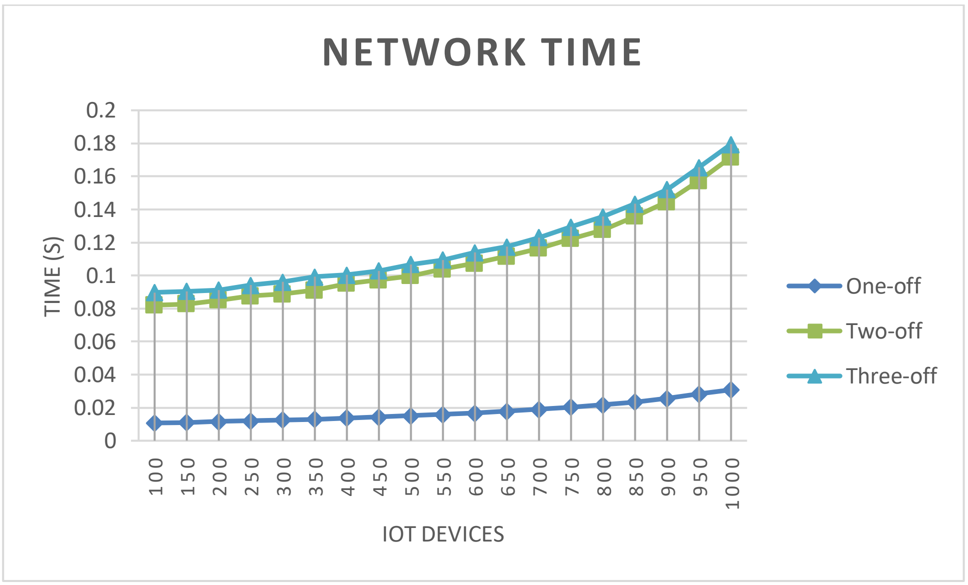


Figure3. 6: Network time for three offloading scenarios

The conducted experiments have shown a significant difference in network time between one level offloading and the others (two-level and three-levels). As mentioned earlier, this is due to the avoidance of WAN and MAN delays. Two offloading levels lower than the three offloading levels because of the further communications between edge nodes.

In term of processing time, as depicted in figure 3.7, offloading to the edge, and the cloud has the lowest service time comparing to others. The reason is that the local edge has limited computational resources; thus if the number of IoT increase it, the processing delays will increase due to limited capacity. On the other hand, offloading to multiple edge nodes with the cloud has the highest processing time. However, the result of processing time was not very encouraging, but in the next chapter, more investigation will be held on the impact of the parameter of processing time (computational demand).

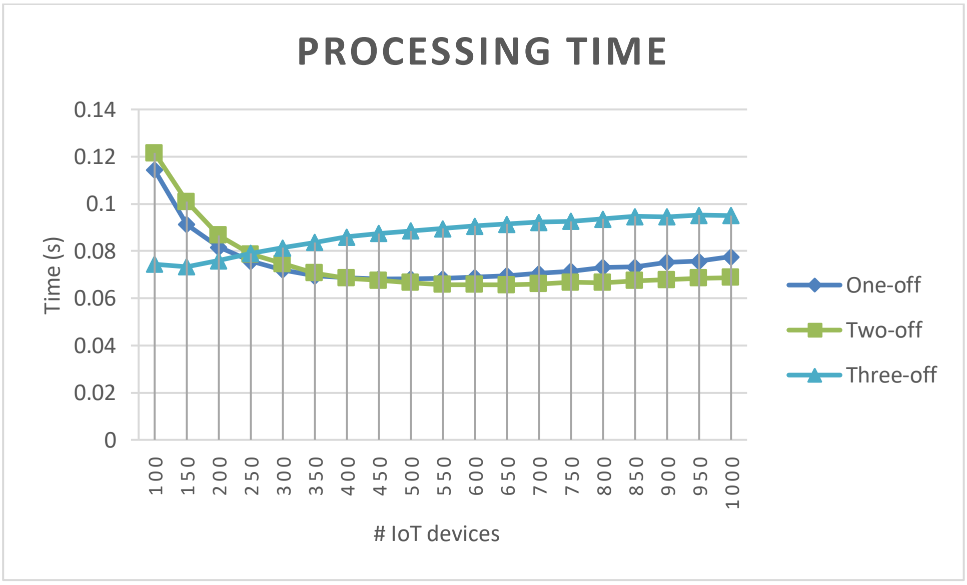


Figure3. 7: Processing time for three offloading scenarios

3.7 Summary

This Chapter has presented the Edge-Cloud system architecture that supports scheduling offloading tasks of IoT application, followed by the explanation of the required components and their interactions within the system architecture.

Moreover, A number of simulation experiments have been conducted in order to investigate the latency of three different offloading schemes. In the next chapter, the factors of latency will be presented and discussed in detail.

Chapter 4. Detail Analysis of the Main Factors of Service Latency

4.1 Overview

This chapter presents the main factors of service latency that will consider the proposed approach for Edge-Cloud resource management. Since the demand for computation and communication tasks vary in IoT applications, this chapter aims to validate the impact of these factors on the overall application latency. Moreover, Edge-Cloud environment consists of heterogeneity of computing resources; thus, selecting the appropriate resources to process the offloading tasks play a critical role to improve the overall service time. Therefore, a number of simulation experiments were set-up to evaluate the effect of these factors.

4.2 Factors of Service Latency

It is advantageous to understand and investigate the precise operational scenarios and factors that affect the overall service time for IoT applications. This is crucial in order to focus technical and developmental efforts toward proposing efficient resource management for Edge-Cloud system. Section 3.4 presented the architectural service delay models in the Edge-Cloud system, which based on the location of the offloaded task (e.g. edge, other collaborative edge or cloud). This Chapter details the main factors that affect the overall service time of latency-sensitive applications and should be considered by the resource manager.

There are several factors, from the application perspective affecting the decision of offloading. For example, for latency-sensitive application could lead to a significant delay in computation and communication. In the following some of these main factors:

1- Application characteristics: this refers to when there are some tasks that are working jointly together, such as Direct Acyclic Graph (DAG). In this case, if the resource manager offloads the tasks in different locations, then it will lead to increase the communication time [117][118]. Therefore, considering the impact of task dependency could improve the application QoS.

2- Application tasks' variation: in general, any IoT application consists of several tasks, and tasks are varied in their functionality; thus, their demands will be different in term of the required CPU or the amount of transferred data. Consequently, it has a significant effect on where to offload the task, which will affect the service time.

3- Types of computational resources: Edge-Cloud resources consist of heterogeneity resources, either with different hardware capabilities or different hardware architecture (e.g. GPU and FPGA). Therefore, the resource manager needs to have an effective approach to manage these resources and assign the tasks to the most appropriate hardware to get the best performance out of these resources.

4- Users mobility: Since some IoT applications require mobility support, thus it might occur when the task offloaded to a local edge node while the IoT device moves to another area that covered by another edge node. Consequently, it could lead to a significant degradation in service time performance [52].

In this work, we particularly target the impact of task's variation in term of computational and communication demands of IoT latency-sensitive applications as well as the offloading to the heterogeneity of computational resources, as presented in figure 4.1.

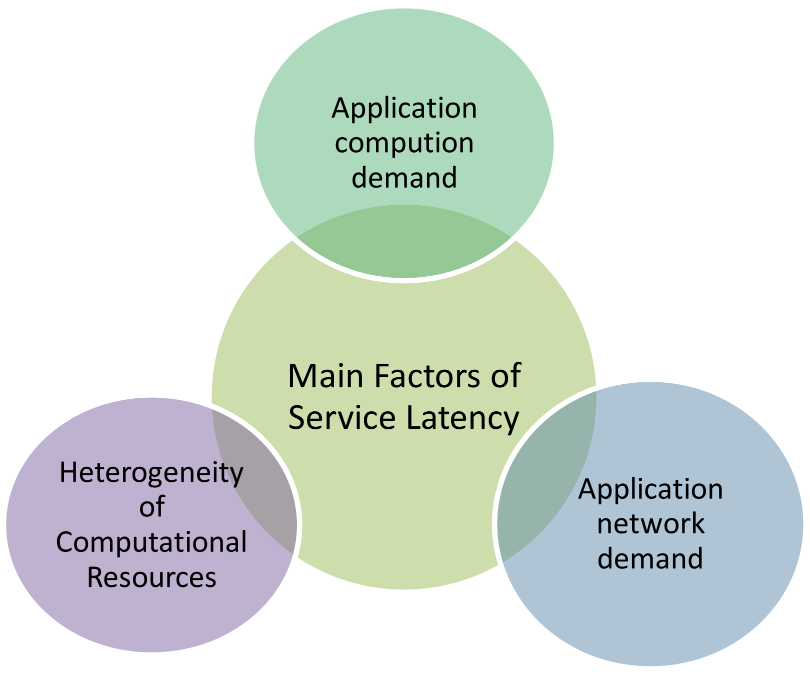

Application compution demand
Main Factors of Service Latency
Heterogeneity of Computational Resources
Application network demand

Figure 4. 1: Main factors of service latency

4.3 Application Characteristics (Computational and Communication)

Tasks of IoT applications can be characterised by its needs for computational resources, (i.e. CPU and RAM) as well as communication needs (e.g. uploading and downloading data). IoT offloaded tasks usually vary the degree of resource reliance between light and heavy. Namely, it ranges from low computation, and communication demands such as health-care applications[119] to high computation and communication demand such as online video gaming [120]. As depicted in figure 4.2, some tasks require more computation time due to the intensive processing, and others require more network time due to transfer a massive amount of data. Thus, it could affect the process of offloading tasks in order to minimise the overall service time.

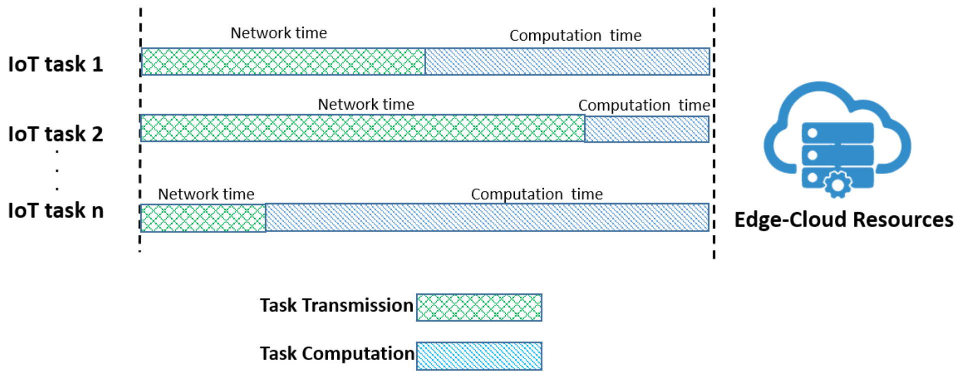


Figure 4. 2: Tasks' variation (computation and communication)

Tasks of IoT latency-sensitive applications, that require higher computational demands prefer to be processed in the cloud since the edge resources are limited, but this also depends on the request of the transferred data. Tasks that require to move a large amount of data needs to be processed in the edge to avoid the long latency in the network backhaul.

Basically, the task completion time consists of three essential components, i.e. computation time, network time and where the task is scheduled (e.g. which server type). The server can be located on the local edge, other nearby edge nodes close to IoT devices or belong to the cloud.

The computation time of the IoT task depends on the number of instructions (e.g. MI) that need to be executed and the processing speed of the hosted resources (e.g. VM). The number of instructions represents the computational volume of an IoT task. As mentioned above, IoT tasks can range from a small number of code instructions to a high number, which depends on the IoT application.

This factor alongside with network conditions specifies where to offload the tasks. For example, it is not logical to offload the tasks with a massive amount of data to the cloud, whereas the edge resources are available, because it will increase the overall service time. However, these two factors come together; thus, we need to understand and investigate the impact of each of them in an independent way. The network time of IoT task depends on the amount of data to be uploaded and downloaded as well as the transmission latency between the sender and the receiver. In our case between the sender will be the IoT devices and the receiver cloud be (local edge, other collaborative edges or the cloud). Moreover, for each task, the amount of transferred data can vary based on the IoT application.

4.3.1 Computational Resource Heterogeneity

In term of computational resources, both IoT devices and edge servers are heterogeneous. Consequently, for latency-sensitive applications, underestimating the computational resource needed for executing the task in order to minimise the overall service time [73]. Due to resource heterogeneity, this means there are some servers that are better than others in term of capabilities, which can handle the offloading tasks faster as presented in figure 4.3. Whereas, the amount of required computational is varying for each task. Thus, heavy tasks required a powerful machine to process their jobs faster.

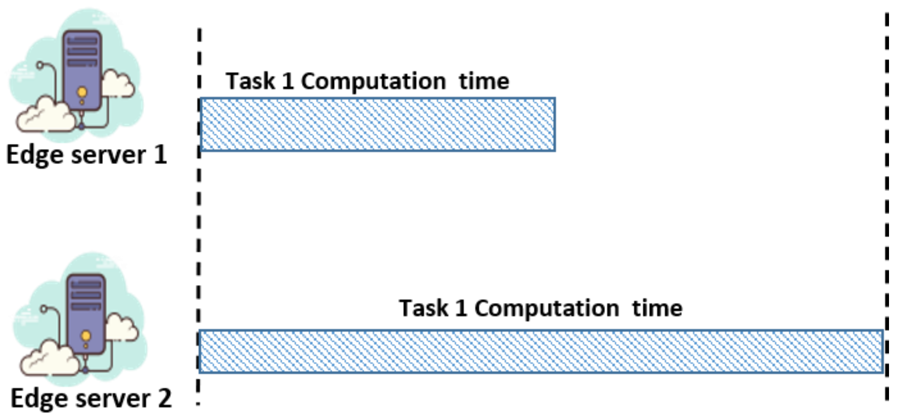


Figure 4. 3: Computation time for resource heterogeneity

Therefore, a performance method to measure the end-to-end IoT service effectiveness, taking both computational and communication demands of offloading tasks into account are needed, in order to answer the following questions:

- How different applications parameters, including computation and communication demands, impact on the overall service time?
- How different computational resources (e.g. different VM capabilities) impact on the overall service time?

4.4 Implementation

In order to understand and investigate the impact of computational and communication demand of IoT tasks as well as the impact of different computational resource capabilities. A number of simulation experiments have been conducted on the EdgeCloudSim (see section 2.6) to mimic Edge-Cloud System. These experiments use different IoT tasks with ranges of computational and communication demands (e.g. different tasks length in MI with different amount of uploading and downloading data in MB).

Characterisation of Virtual Machines

In order to investigate the impact of resource heterogeneity, two different VMs on EdgeCloudSim have been considered. Table 4.1 shows the configurations of the VMs that were considered in the experiments. This based on Rackspace, which provides a wide range of VM types [151] and other works in [88][121] are used as a reference for the VMs configurations. The first type of VM has two cores Intel Xen CPU, and the second type of VM has four cores Intel Xen CPU.

Table 4. 1: Configurations of VMs

	CPU core	**MIPS**	**RAM (GB)**	**Storage (GB)**
VM type 1	2	10000	2000	50000
VM type 2	4	20000	4000	100000

4.4.1 Experimental Investigation

The overall aim of the experiments is to investigate and understand the impact of the change in the computational and communication demands of the IoT tasks as well as the effectiveness of resource heterogeneity in term of the overall end to end service time. Several simulation experiments have been conducted using different IoT offloaded tasks. The simulation key parameters are represented in table 4.2, which contain the number of IoT devices, the number of edge nodes and number of VMs. To mimic various applications that might be encountered in practice, we define the configuration of tasks varying communication bandwidth demand from 0.25MB to 1MB as an increased step of 0.25MB and doubling computation demand starting from 500 MIPS to 4000 MIPS. These numbers have been used in similar related work in the literature to represent offloaded tasks [80]. Also, we did a sensitive analysis of the selected parameters similar to the work in [122]. First, we maintain tasks communication as a constant parameter and vary the task's computational demand to study the impact of the computational demand. Then, increased the communication demand while the computational demand is constant to investigate the communication demand. The impact of computational demand and communication demands are presented in Figure 4.4 and Figure 4.5, respectively. Moreover, we run the same IoT workload with two different VMs, as presented in table 4.1.

Table 4. 2: key parameters of the simulation environment

Parameters	**Values**
Simulation Time	30 minutes
Warm-up Period	3 minutes
Number of Iterations	5
Number of IoT devices	100-1000
Number of Edge Nodes	3
Number of VM per edge server	8

4.4.2 Results

This section investigates and analyses how the size of the demanded resources (CPU and network) of IoT tasks influence the overall service time under various IoT devices. In order to measure the computational demands, we fixed the amount of communication demand and tried different types of task computational and vice versa for the task communication demand. Furthermore, validates the service time performance and utilisation for two different VMs.

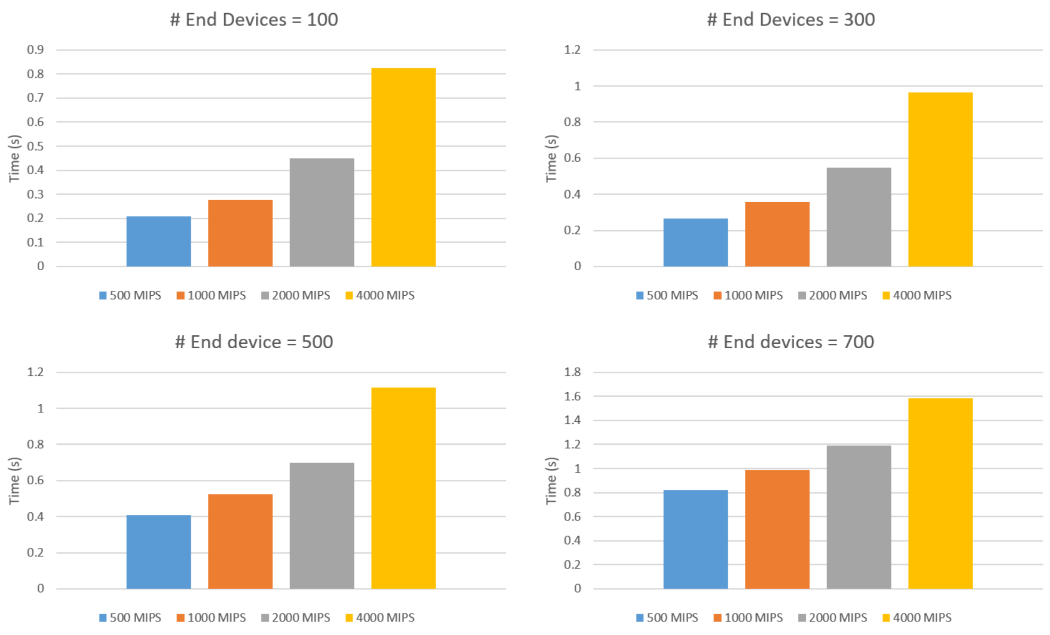


Figure 4. 4: The impact of computation demand

Figure 4.4 shows the average service time for offloading tasks with different computational demand (e.g. 500 MIPS, 1K MIPS, 2K MIPS and 4K MIPS) under a different number of IoT devices. As depicted in figure 4.4, no matter how many IoT devices, the average service time of IoT applications shows a corresponding increase along with the increment of its CPU requirements, and the fewer end devices, the more obvious fluctuation. For example, the end to end service time of 4K MIPS task is about four times of task with 500 MIPS when the number of mobile end devices equals to 100, but only nearly two times when the number is 700. Intuitively, the reason is that computation resources are severely limited, and when the demand for CPU increases, the time of waiting and processing in CPU will also rise correspondingly. However, once the number of tasks increases to a certain value, the conflict of CPU resource (means Clock Cycles) will increase slowly as it’s near to the maximum of CPU capacity.

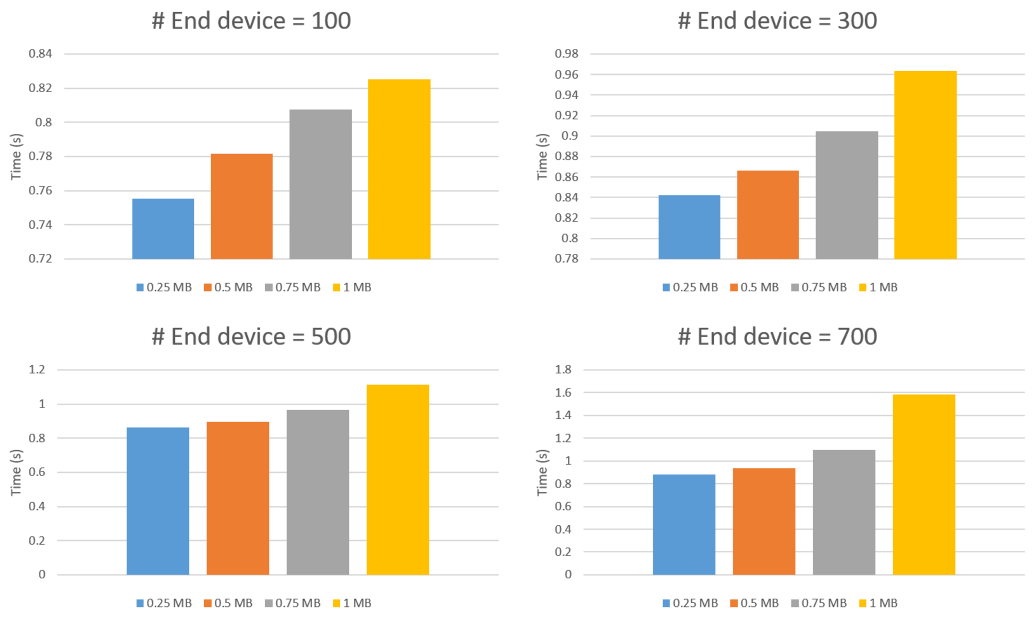


Figure 4. 5: The impact of communication demand

Figure 4.5, shows the average service time for offloading tasks with different communication demand (e.g. 0.25 MB, 0.5 MB, 0.75 MB and 1 MB) under a different number of IoT devices. As shown in figure 4.4, when the bandwidth demand of task varies, the service time only slightly increases due that the network bandwidth is not a critical limit for IoT tasks in current experiments. In other words, network resource or performance is notably sufficient to handle nearly all IoT tasks. Notice, when the number of end devices is close to 700, the increment becomes obvious and efficient assignment of network resources will play a meaningful role in end to end service time.

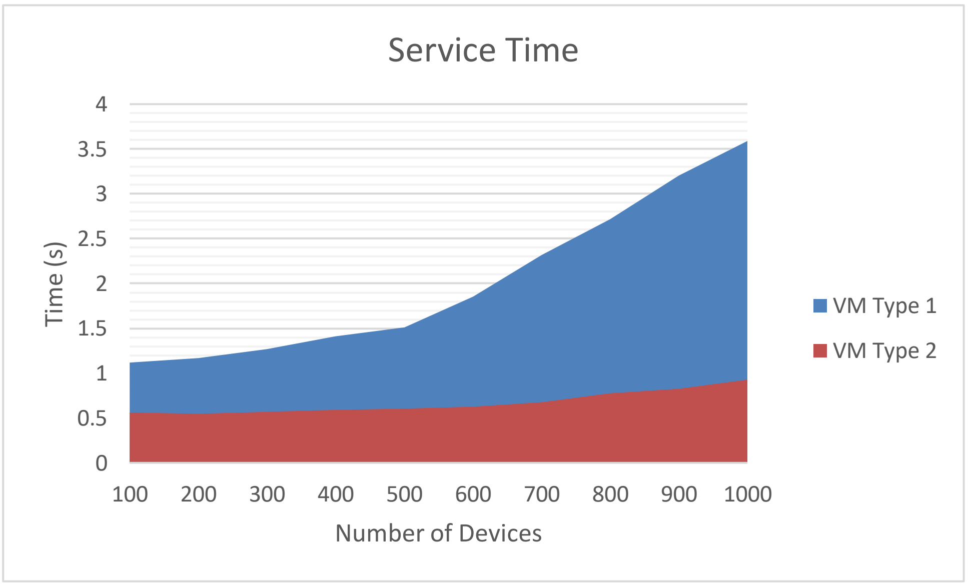


Figure 4. 6: The impact of two different VMs

As the VMs are heterogeneous in terms of size, they consequently have different processing time for IoT offloaded tasks, which fairly corresponds to their size. In the beginning, in figure 4.6, when the number of IoT devices 100, The average service time of IoT offloaded tasks processed in VM type 1 is double comparing the service time in VM type 2. Further, the conducted experiments had revealed that a huge increase in service time when the number of IoT increased for the VM with low capabilities while the VM type 2 handling the increased of IoT devices effectively.

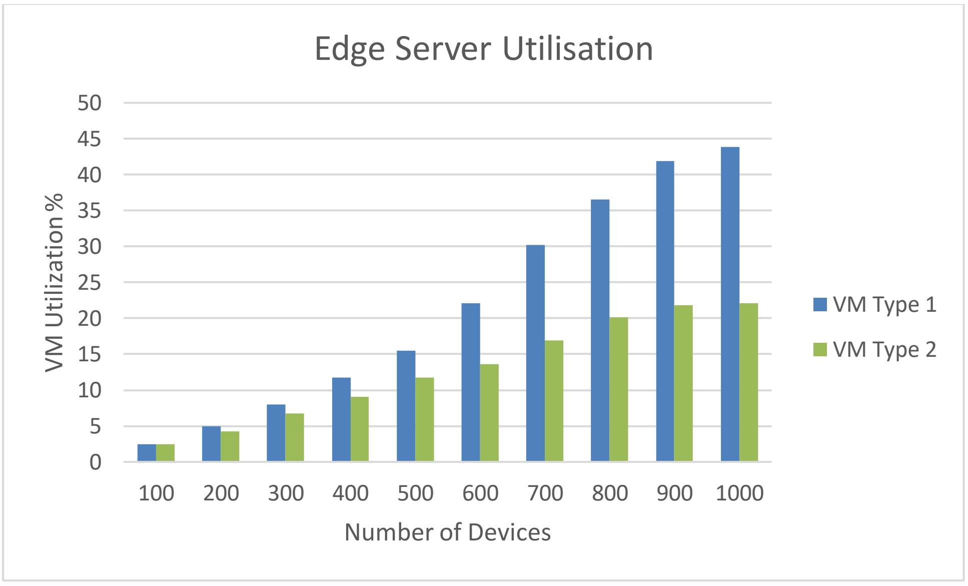


Figure 4. 7: Server Utilisation of two different VMs

As the VMs are heterogeneous, the average utilisation of VMs is consequently different. As shown in figure 4.7, when the number of IoT devices small at 100 devices, both VM type 1 and VM type 2 have the same utilisation level. However, when the number of IoT devices increased around 1000 IoT device, the average server utilisation of a VM type 1 is about twice bigger than a VM type 2. The reason is that VM type 2 has more capability than VM type 1, thus can process more IoT offloaded tasks with an acceptable level of utilisation.

Possible explanations that we can discover it when we are comparing the two results presented in Figure 4.6 and Figure 4.7, there is a correlation between the processing time of the edge server and the utilisation level. Because the trends of both of them have the same level of increase for both VM type 1 and VM Type 2, this effect might be due to the rise of computational load and the IoT devices that sharing the same resources. Also, it possible if we increase the load for VM Type 2, that we get the same results of VM type 1.

4.5 General Discussion

We outline some findings according to our simulation-based evaluation which can be employed in improving the efficiency of task offloading and achieving well-balanced resource management in the Edge-Cloud environment. The conducted experiments on EdgeCloudSim have shown the impact of tasks computational demand and communication demand as well as how it affects the overall service time for IoT application. Some studies [123] emphasise that the central cloud could overwhelm the network delay due to its tremendous resources. One the other hand, other studies [124] show that the increased network demand could lead to an exponential delay for some applications. Based on our simulation results, we show that network time has a significant impact on the overall service time, thus considering this parameter could lead to an improvement in performance. In current experiments, the impact of computational demand affected the overall service time more than the communication demand. For example, when the number of IoT devices equalled 700, the service time increase by around 0.2 seconds when the amount of computational demand increased from 500 MIPS to 1000 MIPS. In contrast, when the amount of communication demand increased from 0.25

MB to 0.5 MB, the overall service time increased by 0.03 second. These results seem to be consistent with other research [125] which found when the Edge-Cloud becomes overloaded, there will be degradation in execution performance due to resources contention and sharing.

Furthermore, the experiments have shown that the measured overall service time for the two types of VMs have a clear impact on the performance when the system load is increased. In accordance with the present results, previous studies on resource heterogeneity have demonstrated that VM diversity results in application performance variation. Based on experimental results conducted on Amazon EC2 large VM can enhance the service time performance up to 40%, and for some specific applications could reach 60% [126]. Moreover, when the server utilisation increased, the overall service time is sharply increased. This finding was also reported by [127] when the amount of computational workload increased, server utilisation will be increased and the service time performance will be affected. This result is expected and the main motivation of this simulation is to demonstrate the necessity of considering the resource utilisation level in the process of scheduling offloading tasks in order to minimise the overall service time.

4.6 Summary

This chapter has presented and illustrated the impact on the performance of IoT applications under different requested resource (CPU and network) changes and examines the effectiveness while varying the number of IoT devices. Furthermore, the impact of resource heterogeneity and server utilisation on the performance IoT application service time.

Chapter 5. New Approach to Task Offloading in an Edge-Cloud Environment

5.1 Overview

In this chapter, a new approach for task offloading in edge-cloud systems is proposed in order to minimise the overall service time for latency-sensitive applications. The approach adopts the fuzzy logic algorithm that considers application characteristics (e.g. CPU demand, network demand and delay sensitivity) as well as resource utilisation and resource heterogeneity, as presented in section 5.2. A number of simulation experiments evaluating the proposed approach in relation to other related works are then presented in sections 5.3 and 5.4.

5.2 Tasks Scheduling Approach with Minimum Latency

In the Edge-Cloud environment, IoT devices produce a stream of incoming offloading tasks that differ in term of their computation and network demand. Generally, task scheduling is used to enhance several performance parameters, which include minimising the overall delay in the processing of offloaded tasks. The companied Edge and Cloud environments consist of a set of heterogeneity resources (e.g. different computation resource capabilities). Therefore, a realistic model of scheduling offloading tasks in Edge-Cloud system consists of several parameters, which can be organised into two main categories: infrastructure characteristics and application characteristics. Infrastructure characteristics include resource heterogeneity which could lead to select the appropriate resource for a specific task; also, the utilisation level of edge server and the network conditions. For example, CPU utilisation could vary depending on the assigned task and depending on whether the number of IoT devices increases in a shared network. This could lead to fluctuations in network bandwidth. On the other hand, IoT application tasks include characteristics such as computation demand, required transfer data for uploading and downloading, and the needed deadline to complete tasks.

From the previous explanation, the problem might be seen as a classic multi-constraint optimisation which can be modelled and solved mathematically. However, the edge-cloud environment changes dynamically and unpredictably. For instance, the number of IoT devices could be increased or decreased in a specific area due to IoT mobility, which has an impact on the load of the edge node and the shared network. Also, the incoming tasks are not known in advance, which requires a system to handle them in real time. Therefore, due to the dynamic change in the demands on IoT applications and in the status of edge-cloud resources, it is difficult to get accurate mathematical models [128] because of uncertainty and vagueness [129]. Several research efforts have addressed the latency challenges in the edge-cloud environment. Researchers in [130][131][77] have considered the computational and communication parameters in order to enhance the overall latency. Besides, several studies [78][132][4] have investigated the impact of resource heterogeneity in the edge-cloud environment and its role in enhancing the end-to-end service time. Also, a considerable amount of literature [133][134][135] has been published on load balancing and server utilisation in edge-cloud systems in an effort to avoid overloaded edge nodes which affect application service time. Overall, these studies highlight the need for an approach that considers the above parameters in term of application characteristics (computational, communications and latency), resource heterogeneity and resource utilisation in order to reduce overall latency and enhance resource utilisation.

This is known as a dynamic multi-objective optimisation problem, where there is more than one objective and the parameters are dynamic in nature and can vary over time [136]. Most of the studies in the edge-cloud area are designed to meet a specific scenario or for a particular application, which makes them less adaptive and scalable [106]. Also, the complexity and the amount of time needed for solving this problem should be considered, since the resources at the edge involve computational constraints. Thus, solving this problem with traditional methods at the edge nodes could add extra overhead, which affects the ability to meet stringent service requirements in latency-sensitive applications [136].

Therefore, fuzzy logic is considered to be among the most feasible solutions for a multi-objective optimisation problem when the activity of multiple parameters is significant. Fuzzy logic can be easily adapted to the dynamicity of computational resources and application parameters as well as providing scalability within the context of the system. It also averts the computational complexity and can provide decisions very quickly [137]. Consequently, fuzzy logic has been adopted in this research to determine where to offload the tasks based on application and system parameters. Many researchers in the field of the distributed systems use fuzzy logic to deal with the challenges caused by vagueness, uncertainty and the dynamicity of the environment [138]. The concept of fuzzy logic is to abstract the problem complexity to a level that can be understood. It can handle system uncertainty by dealing with many input and output variables and can represent the problem with simple if-then rules. To the best of our knowledge, this is one of the early attempts to design and implement such a system with regards to application's demands, edge-cloud resource utilisation and resource heterogeneity by adopting Fuzzy logic. The main objectives are to reduce the overall service time and to utilise the Edge-Cloud resources efficiently.

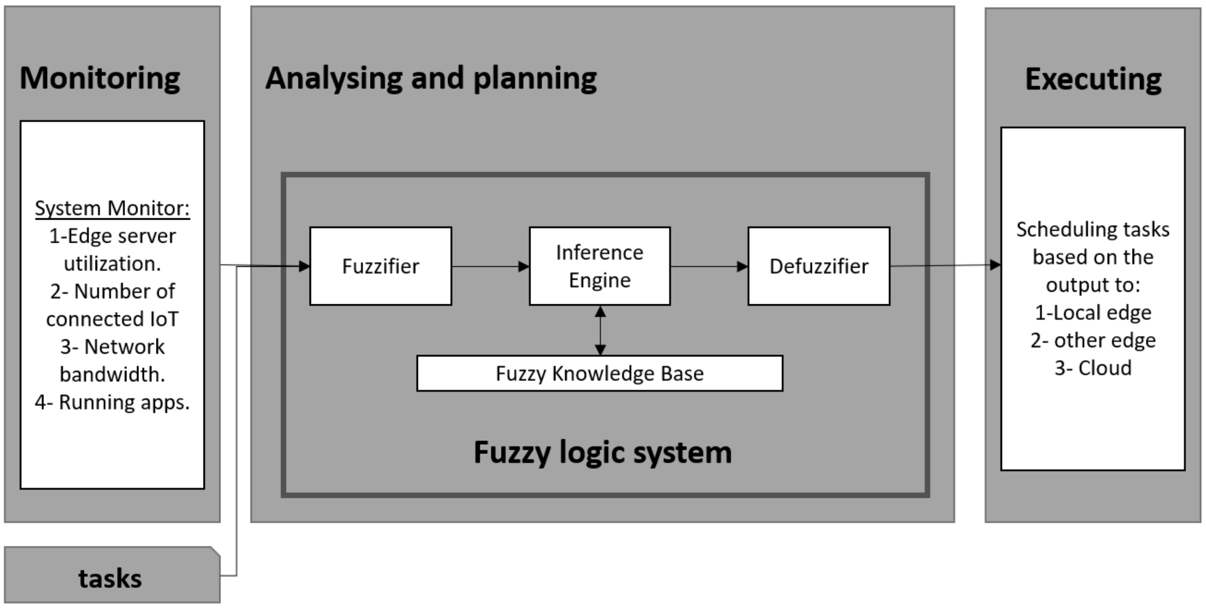


Figure 5. 1: The proposed approach of scheduling offloading tasks

The proposed approach supports the resource manager in the Edge-Cloud system regarding scheduling the offloading tasks in order to minimise the overall service time and improve the efficiency of edge-cloud resources. As shown in Figure 5.1, the approach can be described using the MAPE method (monitoring, analysing, planning and executing) to assign the tasks to appropriate resources and monitoring the system performance periodically. The proposed approach will work in the EC, which presented in Figure 3.1 in Chapter 3. EC is an independent entity in the edge layer that mange all the edge nodes and also responsible for receiving/scheduling the offloading tasks in order to satisfy applications' users and Edge-Cloud System requirements. Therefore, after the edge-cloud system receiving the offloading tasks, the system gets the required information from monitoring data such server utilisation and then pass this information to the fuzzy logic system in the analysing and planning phase. Then the tasks will be scheduled to the appropriate resources based on algorithm 2.

5.2.1 Fuzzy Logic System

In this stage, the proposed approach will get the information of the offloading tasks and server utilisation in order to determine the appropriate location of the offloading tasks as depicted in figure 5.2. The following is a brief description of the process of the fuzzy logic system.

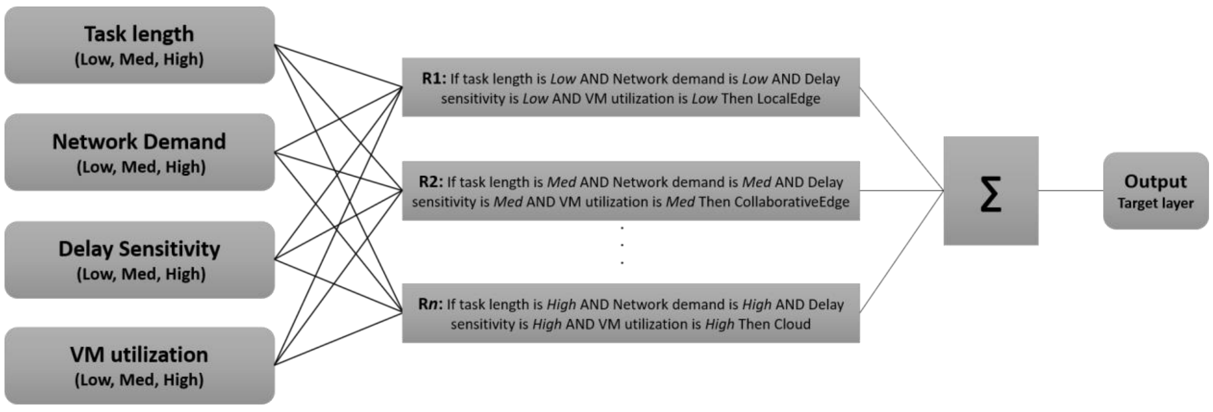

Task length
(Low, Med, High)
Network Demand
(Low, Med, High)
Delay Sensitivity
(Low, Med, High)
VM utilization
(Low, Med, High)
R1: If task length is Low AND Network demand is Low AND Delay sensitivity is Low AND VM utilization is Low Then LocalEdge
R2: If task length is Med AND Network demand is Med AND Delay sensitivity is Med AND VM utilization is Med Then CollaborativeEdge
Rn: If task length is High AND Network demand is High AND Delay sensitivity is High AND VM utilization is High Then Cloud
Σ
Output
Target layer

Figure 5. 2: Process of the proposed fuzzy logic system

1- **Fuzzy input variables:** In this step, we specify the necessary inputs for the fuzzy system. The required inputs are VM utilisation at the edge, task length, the amount of data to be transferred for each task and delay sensitivity. All these variables are represented as a linguistic variable: Low, Medium and High as depicted in figure 5.2. These categories represent the dynamic changing over Edge-Cloud infrastructure and the characteristics of applications' offloaded tasks.

   a) VM utilisation: This parameter indicates the current utilisation level of the VM hosted by the local edge server. Thus, we can know how much resource space is available on that VM. If it is highly utilised, then offloading to other edge servers or the cloud could be the solution, depending on the task characteristics in term of computational, communication and latency sensitivity.

   b) Task length: this parameter represents the computational demand of the task; it measures by Million instruction per Second (MIPS). As the edge has a limited computational resource, heavy tasks might be appropriate to offloaded to the cloud and vice versa. However, we cannot take this parameter without considering others such as VM utilisation, communication demand and delay sensitivity.

   c) Network Demand: This parameter represents the required communication of the tasks for both uploading and downloading. It is an important measure for the offloading decision to consider where to offload the task to (local edge, other edge or cloud).

For example, tasks of Augmented Reality applications that require video streaming must upload the request, then do some processing (e.g. 3D rendering, image processing, etc.), and then receive the results as a video stream. This requires transferring a high amount of data for uploading and downloading, which takes a significant amount of the total service time.

d) Delay sensitivity of the task: This parameter refers to the sensitivity of the tasks to accept the delay due to computation delay or communication delay. For example, some application has some urgent tasks that require ultra-low latency and some tasks that can accept some higher level of latency. This parameter could help task scheduler to assign the tasks to an appropriate server within the Edge-Cloud system.

2- **Fuzzification:** In the fuzzification stage, fuzzifier will take all the required values as numerical input from system infrastructure monitoring and incoming tasks. Then, assign each value to its predefined linguistic variables in the membership functions (e.g. Low, Medium and High). After that, fuzzy variables are combined and evaluates in the Fuzzy rules base to take the decision and produce the output in the defuzzification stage.

a) Fuzzy membership functions: The fuzzy membership function is used to quantify the linguistic term for each fuzzy variable. In this research, we have four functions and each function has three variables: Average VM utilisation (Low, Medium, High), Task length (Low, Medium, High), network bandwidth (Low, Medium, High) and Delay sensitivity (Low, Medium, High). The values of each fuzzy variables are determined empirically based on a number of experiments similar to researches on[139] [85]. Figure 5.3 shows the four membership functions.

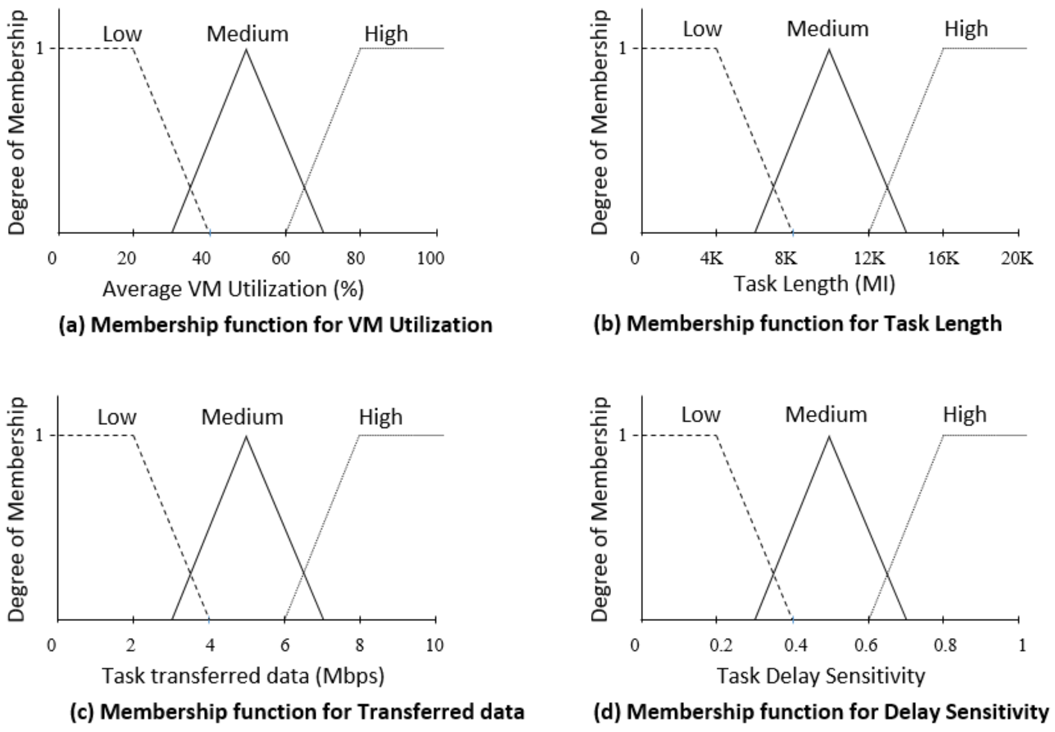


Figure 5. 3: Memberships functions of the proposed fuzzy logic system

b) Fuzzy rules base: A fuzzy rules base is composed of a set of fuzzy rules that similar to the reasoning process of human. It is a simple if-then rule that covers all the possible situations of the application characteristics and system conditions. These rules play critical rules to define the overall system performance. An example of the rules, if task length is *high* AND Network demand is *low* AND VM utilisation is *high* AND the delay sensitivity is *high* THEN offloaded the task to the cloud. The output will be used in the defuzzification stage. Table 5.1 gives results examples of the system fuzzy rules. The main aim is to provide low latency for IoT applications by reducing the data movement from IoT device to the cloud and avoiding the overloaded node, which will affect the end to end service time.

Table 5. 1: Fuzzy rules base

Input variables				**Output Decision**
Task length (MIPS)	**network Demand (Mbps)**	**VM utilisation**	**Delay sensitivity**	
Low	Low	Low	Low	Local Edge
Low	Low	Low	Medium	Local Edge
Low	Low	Medium	High	Local Edge
Low	Low	Medium	Low	Local Edge
Low	Medium	High	Medium	Local Edge
Low	Medium	High	High	Local Edge
Low	Medium	Low	Low	Local Edge
Low	Medium	Low	Medium	Local Edge
Medium	High	Medium	High	Other Edge
Medium	High	Medium	Low	Other Edge
Medium	High	High	Medium	Other Edge
Medium	High	High	High	Other Edge
Medium	Low	Low	Low	Local Edge
Medium	Low	Low	Medium	Other Edge
Medium	Low	Medium	High	Other Edge
Medium	Low	Medium	Low	Other Edge
High	Medium	High	Medium	Cloud
High	Medium	High	High	Cloud

High	Medium	Low	Low	Other Edge
High	Medium	Low	Medium	Other Edge
High	High	Medium	High	Cloud
High	High	Medium	Low	Cloud
High	High	High	Medium	Cloud
High	High	High	High	Cloud

3- **Defuzzification:** Defuzzification is the process to convert the fuzzy rules output to a specific value based on the output membership function. There are a range of ways to produce the output membership function in the fuzzy logic system and these examples of the often-used method (e.g. Maximum, Mean of Maximum and centroid). This work adopts the maximum approach because our membership function has one maximum at a time. Figure 5.4 represents the output membership function of the fuzzy logic system. For example, if the output fuzzification process is 38, then $\mu^{Local\ Edge}$ is 0.1 and $\mu^{Collaborative\ Edge}$ is 0.4, the defuzzification process will take the maximum, and the task will be offloaded to the other collaborative edge node.

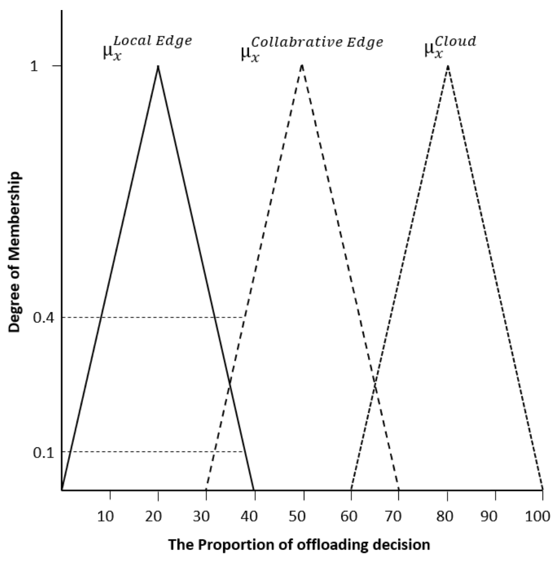


Figure 5. 4: The output membership function of the fuzzy logic system

Algorithm 1 present the process of the proposed fuzzy logic system and the possible outputs. It gets all the required inputs from the offloading tasks of multi-user (e.g. CPU length, network, and delay) as well as Edge-Cloud resources (e.g. VM utilisation). Line 3 calls the fuzzy logic function and pass the needed parameters, as described above. Then the offloading task is allocated based on the output of the fuzzy logic system to one of the edge nodes or the cloud.

Algorithm 1 Fuzzy Logic for offloading to the target layer

INPUT: Applications' tasks T_i with their parameters T_{length},$T_{Network}$ and T_{Delay}

OUTPUT: Select computational resources at the target layer: R_{Local_Edge}, $R_{Collaborative_Edge}$ and R_{Cloud}

```
1: for all Tasks in $T_i$ do
2:     Read VM average utilization $VM_u$
3:     $F$ = FuzzyLogicSystem($T_i^{length}$,$T_i^{Network}$,$T_i^{Delay}$ ,$VM_u$);
4:     if ($F \leq F_{Low}$ ) then
5:         Allocate $T_i$ on $R_{Local_Edge}$
6:     else if $F \leq F_{Med}$ then
7:         Allocate $T_i$ on $R_{Collaborative_Edge}$
8:     else
9:         Allocate $T_i$ on $R_{Cloud}$
10:    end if
11: end for
```

The overall time complexity of Algorithm 1 is O(t) where t denotes the number of elements in T. The step of sending the required information to the fuzzy logic system for each task requires O(t) time. According to the fuzzy inference logic, one of the three different output of fuzzy sets can be allocated to each task; thus, the time complexity of proposed fuzzy logic is O(n).

5.2.2 Task Selection Phase Based on The Resource Type

As shown in Figure 5.5, the incoming tasks inter to the fuzzy logic system. Then after the proposed fuzzy logic system, decide the target layer to offload the task, task scheduling algorithm will assign the tasks to the appropriate computational resources within Local Edge or Collaborative edge based on the information from Infrastructure monitoring. This process runs on the edge controller, which described in Chapter 3. We assume that each Edge node has a heterogeneity of computational resources. The details of the algorithm 2 are described below.

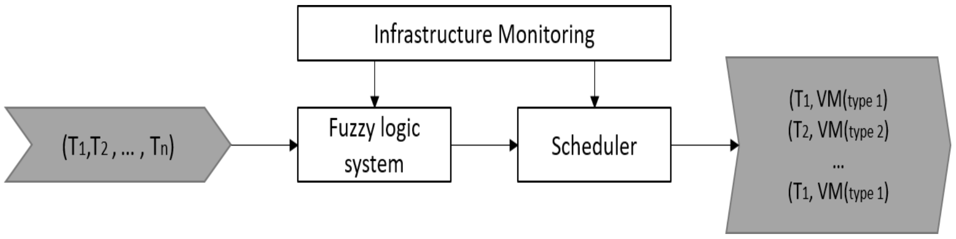


Figure 5. 5: Task selection phase

All the application tasks are submitted to the edge control node in the target layer. Firstly, the algorithm will sort all the tasks in descending order based on their CPU requirements; thus, heavy tasks comes first (Line 1). Secondly, sort all the computational resources (VMs) in descending order based on their CPU capabilities; thus, the most powerful VMs comes first (Line 2). After that, the algorithm will assign each application tasks to computational resources. It will start with heavy tasks to be assigned to the powerful VMs (Line 3-11). This ensures that heavy tasks have the priority to be assigned to a powerful VM, thus will produce less processing time.

```
Algorithm 2 Task Scheduling Algorithm
INPUT: Set of Applications $A_i$, Set of Tasks $T_j$ and Set of Computational Resources at Edge node $R_c$
OUTPUT: Assign Task $T_{ij}$ to Target Computational resources $R_c$
 1: Sort task list $T$ in descending order of task length.
 2: Sort computational resources list $R$ in descending order of resource capacity.
 3: for all Application in $A_i$ do
 4:    while $T_{ij}$ has not been scheduled do
 5:       if ($T_{ij}^{CPU} \leq R_c^{CPU}$ ) then
 6:          Assign $T_{ij}$ on $R_c$
 7:          $T_{ij}$ ++
 8:       else
 9:          $R_c$ ++
10:       end if
11: end for
```

Algorithm 2 aims to schedule the tasks of each application to the target computational resources in the edge node; therefore, the time complexity can be analysed as follow. The first steps are to sorting the tasks and resources based on their computational demand and capacity respectively. The worst-case time complexity of the sorting is $O(n^2)$. The second step is to allocate each task T for all applications A to computational resources R. The big O notation of the nested loop is O(n*n), the first loop for all the applications and the second loop for all the tasks. As a consequence, the overall time complexity of task scheduling algorithm is $O(n^2)$.

5.3 Implementation

The task offloading approach based on a fuzzy logic system that aims to enhance the end-to-end service time by considering both tasks requirements and resources of Edge-Cloud system has been introduced. In order to evaluate this approach, several experiments have been conducted for the proposed approach and compare it with other competitors' solutions. The process starts with generating tasks of different IoT applications, then scheduling tasks in the Edge-Cloud system based on the scheduling algorithms. Details of IoT tasks and the environment of the experiments will be presented in the following sections.

Approaches that dealt with offloading tasks by using fuzzy logic are limited in the area of edge computing. Therefore, we evaluate the proposed approach with the following algorithms. First is a utilisation-based approach, which makes decisions on offloading tasks based on the server utilisation level, by selecting the least-load machine for offloaded tasks. The aim of this approach is utilising edge resource and make load balancing. This approach has been adopted in a number of studies due to the simplicity of its logic and the feasibility of its implementation [140][141]. It is well suitable for the common situation, in which the number of applications and the execution time of tasks is both moderate. However, it doesn't consider task communication demand and application delay sensitivity. Second, Flroes [142] proposed a task offloading approach based on fuzzy logic for IoT applications. This approach aims to decide whether to offload to the cloud or perform the tasks in end devices at the edge layer. However, this approach neglected the utilisation of the Edge-Cloud resources, which could cause an overloaded VM, thus could lead to significant latency. Finally, Snomes [85] proposed tasks offloading approach that consider both applications tasks requirements and resource utilisation by using a fuzzy logic system. However, this approach focused on homogeneous resources, whereas the Edge-Cloud system is composed of resources heterogeneity. Moreover, their solution decides whether to offload to the Local Edge or the Cloud, whereas our proposed approach considers the heterogeneity of resources as well as the available resources in other nearby edge nodes. All of these approaches have been implemented in the simulation tool in order to evaluate it with the proposed approach.

5.4 Experiments and Evaluation

A number of experiments have been conducted on the EdgeCloudSim. EdgeCloudSim has been used because it provides the vital functionality of Edge-Cloud environment such as support offloading, users mobility etc. Section 2.6 provides more details on EdgeCloudSim and its components. The aim of the experiments is to show that the proposed fuzzy logic approach for minimizing end-to-end service time is capable of considering both applications requirements (e.g. computational, network and delay) and the dynamicity of the edge cloud system in terms of resource utilisation. Moreover, the intention is to evaluate the proposed approach by comparing to other approaches in the field.

5.4.1 Simulation Setup

In the Edge-Cloud environment, there are a number of IoT/ mobile devices that have a number of applications. These applications consist of different tasks which require to be processed in the Edge-Cloud resources. Edge nodes are distributed closer to end devices, and we assume each edge node cover a specific area. IoT devices connect to the nearest edge node through WLAN and then can send the offloaded tasks. We assume that each node has a node manager and all edge nodes are managed by the edge controller described in Chapter 3. In our experiments, we assume that we have three edge nodes and a variable number of IoT devices, from 200 to 2000, dispersed and mobile between the three nodes. Table 5.2 represents the key parameters of our simulations.

Table 5. 2: Simulation key parameters

Parameters	**Values**
Simulation Time	30 minutes
Warm-up Period	3 minutes
Number of Iterations	5
Number of IoT devices	200-2000
Number of Edge Nodes	3
Number of VM per edge server	8
Number of VM in Cloud	Not limited
Probability of selecting location	Equal

We assume that we have heterogeneity of VMs on each edge node. Table 5.3 shows the configurations of the VMs that were considered in the experiments. Rackspace, which provides a wide range of VM types [151] and other works in [88][121] are used as a reference for the VMs configurations. Two types of VMs are used in this thesis with different capabilities to supports the end devices with computational resources.

Table 5. 3: Configurations of VMs

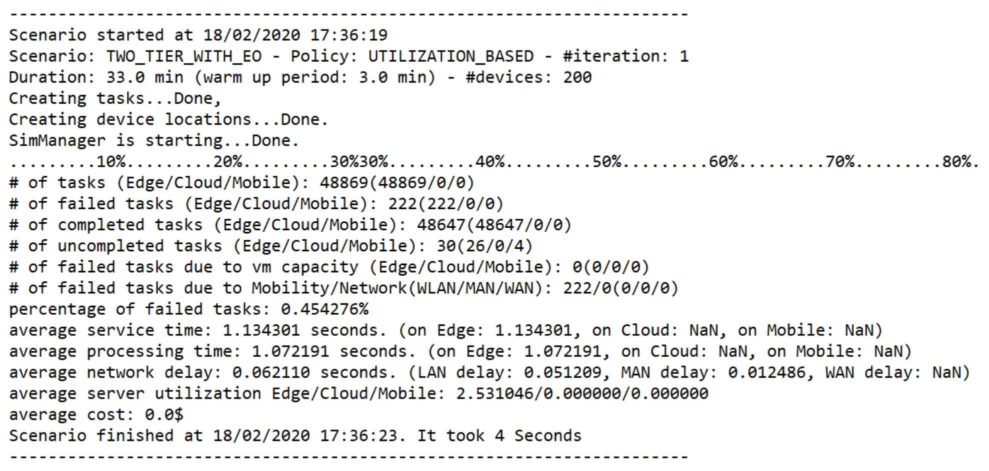
VM type	**CPU cores**	**MIPS**	**Storage**
Medium VM	2 vCPUs	10000	50000
Large VM	4 vCPUs	20000	100000

IoT applications generate different offloading tasks in term of CPU and network load. To evaluate our approach, we need different applications with different computational and communication demands. Several research studies generate random tasks in their experiments [80][143]. Table 5.4 summarised the main characteristics of the four applications that are used in this experiment, similar to [85][88]. Task Length refers to require CPU resources for the task in Million instructions(MI) unit. Uploading and downloading data determines the amount of data to send/receive for each task from the IoT device to the Edge-Cloud system. Delay sensitivity refers to the acceptance level of delay sensitivity.

Table 5. 4: Application characteristics

Apps	**Task length (MI)**	**Uploading Data (KB)**	**Downloading Data (KB)**	**Delay Sensitivity**
Augmented Reality	9000	1500	25	0.9
Health Care	3000	20	1250	0.7
Compute Intensive	45000	2500	200	0.1
Infotainment	15000	25	1000	0.3

Figure 5.6 shows a snapshot of the simulation results for one scenario. Each scenario takes one approach (e.g. Utilisation_Based) with a specific number of devices. We did five iterations for each scenario and took the medium in order to avoid errors. Next section will present the collected results and discussion.



```
------------------------------------------------------------------
Scenario started at 18/02/2020 17:36:19
Scenario: TWO_TIER_WITH_EO - Policy: UTILIZATION_BASED - #iteration: 1
Duration: 33.0 min (warm up period: 3.0 min) - #devices: 200
Creating tasks...Done,
Creating device locations...Done.
SimManager is starting...Done.
.........10%.........20%.........30%30%.........40%.........50%.........60%.........70%.........80%.
# of tasks (Edge/Cloud/Mobile): 48869(48869/0/0)
# of failed tasks (Edge/Cloud/Mobile): 222(222/0/0)
# of completed tasks (Edge/Cloud/Mobile): 48647(48647/0/0)
# of uncompleted tasks (Edge/Cloud/Mobile): 30(26/0/4)
# of failed tasks due to vm capacity (Edge/Cloud/Mobile): 0(0/0/0)
# of failed tasks due to Mobility/Network(WLAN/MAN/WAN): 222/0(0/0/0)
percentage of failed tasks: 0.454276%
average service time: 1.134301 seconds. (on Edge: 1.134301, on Cloud: NaN, on Mobile: NaN)
average processing time: 1.072191 seconds. (on Edge: 1.072191, on Cloud: NaN, on Mobile: NaN)
average network delay: 0.062110 seconds. (LAN delay: 0.051209, MAN delay: 0.012486, WAN delay: NaN)
average server utilization Edge/Cloud/Mobile: 2.531046/0.000000/0.000000
average cost: 0.0$
Scenario finished at 18/02/2020 17:36:23. It took 4 Seconds
------------------------------------------------------------------
```

Figure 5. 6: A snapshot of the simulation results for one scenario

5.5 Results

This section presents the quantitative evaluation of the proposed approach compared to other related works' algorithms (e.g. utilisation-based, Sonmez and Flores). The simulation results consist of the average service time, average processing delay, average network delay, average tasks failure and average VM utilisation. The service time of each task will depend on the location of processing, which can be one of the following: 1) Local Edge, the overall service time consist of WLAN time and processing time. 2) Collaborative Edge, the overall service time will be WLAN/MAN time and processing time. 3) Cloud, the overall time consist of WLAN/MAN/WAN time and processing time in the cloud. After that, we take the average for all tasks in each scenario see the following equation.

$$Service\ Time = \frac{\sum T_{processing\_time} + \sum T_{network\_time}}{Number\ of\ Tasks}$$

The main performance metric is the service time since the end-to-end service time of an offloading task is most significant for IoT latency-sensitive application. The average service time of our approach with other related works algorithms is shown in figure 5.6. It composed of processing time and network time. The purpose of Experiments was to enhance the resources management in Edge-Cloud system in order to reduce latency for IoT applications. As shown in figure 5.6, the proposed approach and other related algorithms have nearly the same performance when the system unloaded. Yet, when the number of IoT devices increases, the number of offloaded task increase, the service time of the proposed approach remain steady compared to others. The chart shows that there has been a sharp rise in Flores algorithm after the number of IoT devices increased more than 1400. The Utilisation_based and Sonmez algorithms nearly have the same performance. Figure 5.8 compares the average network time of all algorithms. It can be seen that all the algorithms have the same time when the system stable, but after it becomes overloaded, they differ. It is obvious that the proposed approach does not have the lowest network time. The utilisation-based approach provides the lowest network time comparing to other approaches.

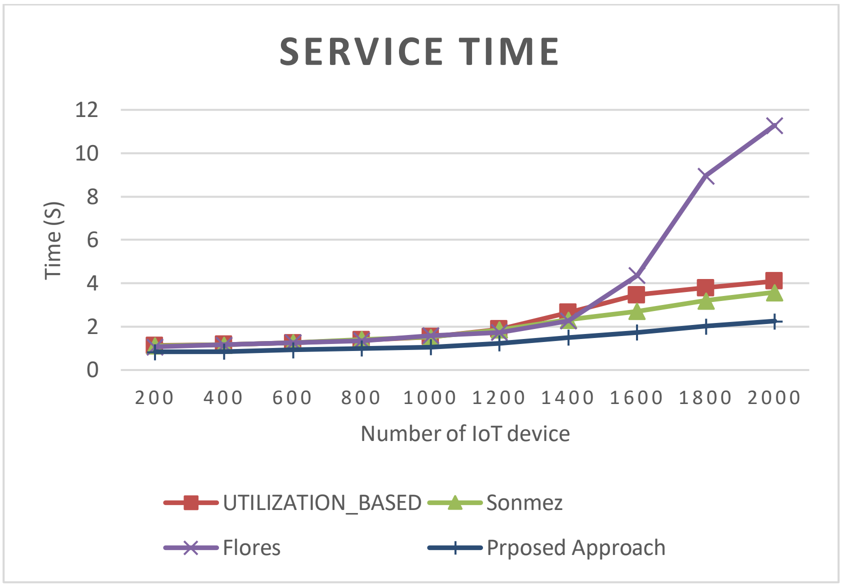


Figure 5. 7: The service time of the proposed approach with other related approaches

Prior studies [28][106] have noted the importance of considering the heterogeneity of resources in the process of task scheduling to reduce the latency. Meeting this demand enables the proposed approach to achieve a better performance than the others. The proposed approach considers the VM utilisation in the offloading decision; thus, the task might be sent to the cloud, which could increase the network time. The observed difference between the proposed approach and Flores in network time might be explained in this way: Flores offloads the task to the edge whenever possible without considering whether the resource is overloaded. This led to an increase in the processing time, but the proposed approach takes consideration of VM utilisation in order to avoid processing delay due to overloaded VM.

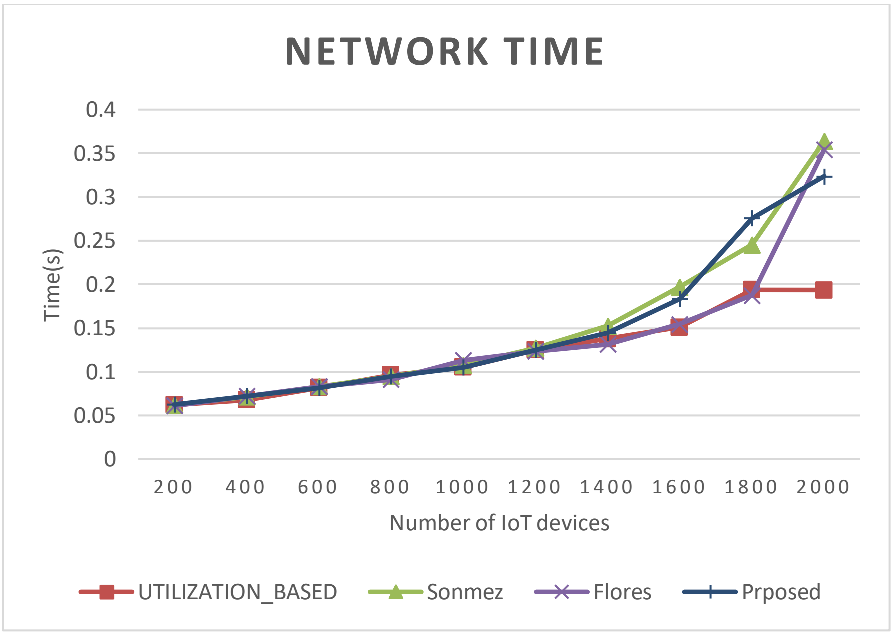


Figure 5. 8:The network time of the proposed approach with other related approaches

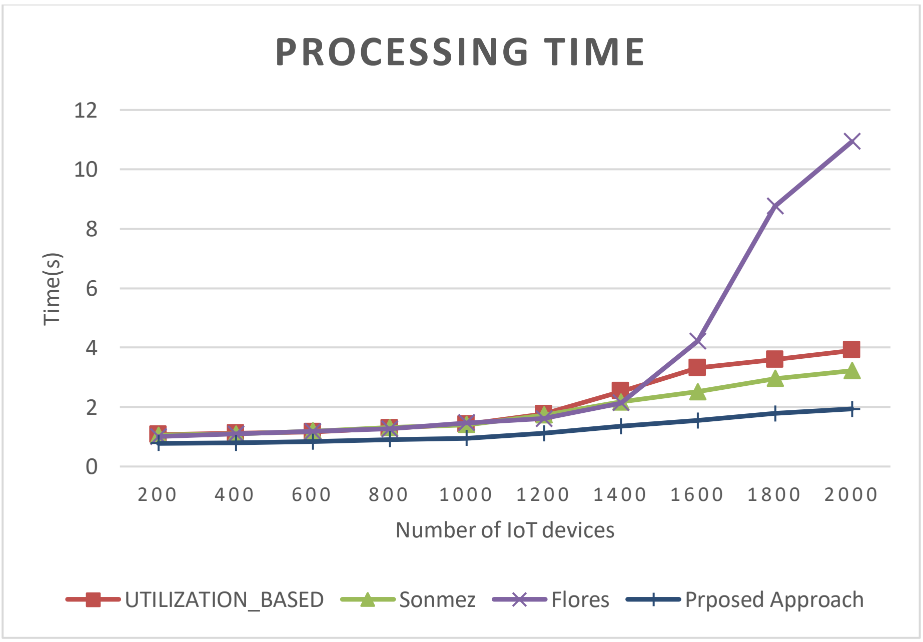


Figure 5. 9: The processing time of the proposed approach with other related approaches

Based on EdgeCloudSim, tasks can fail due to various reasons such as the lake of computational resources in VM (e.g. overloaded VM) and congested network. Therefore, it is an important performance metric that should consider in order to evaluate the offloading approach. The results of task failures can be divided into two parts, when the system stable and when the system overloaded. For the first part, as shown in figure 5.10, all the algorithms have nearly the same performance; around 0.5% of tasks will fail. The proposed approach has the lowest percentage because it considers the required amount of data to be uploaded and downloaded. On the other hand, as shown in figure 5.11, when the system load is high, it can be seen that by far, the lowest task failure average is for the proposed approach. Interestingly, there were slight differences between Utilisation_based and Flores for all number of IoT devices. When the system load is low, most of tasks failures due to network issues such as losses of the packet [6] but when the system overloaded failures can happen because of the lack of computation (e.g. unsuccessful completion task) as well as network issues. The proposed approach was the lowest because it assigns the heavy tasks to the powerful VM as well as considering the other factors (e.g. VM utilisation, network demand and delay sensitivity).

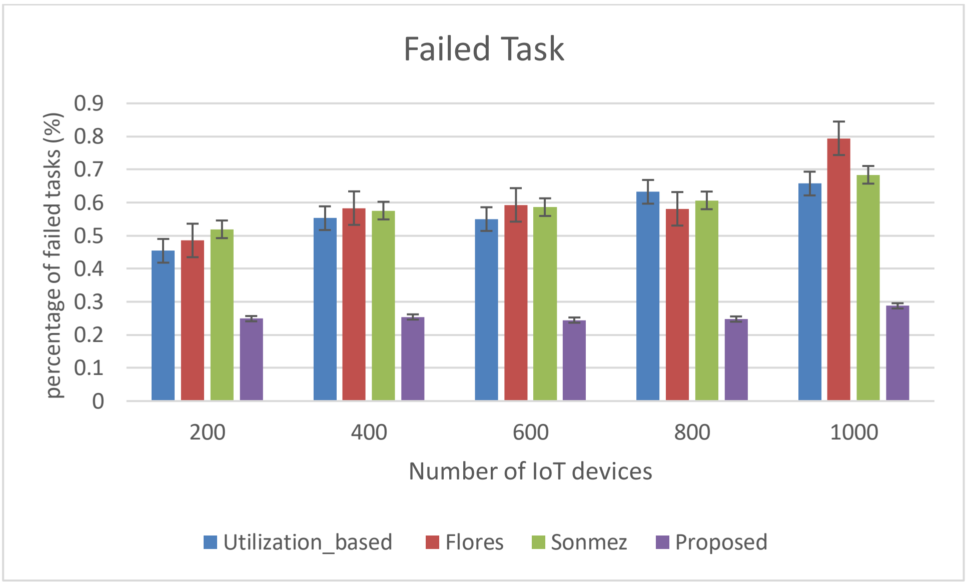


Figure 5. 10: Percentage of failed tasks of the proposed approach with other related approaches Part 1

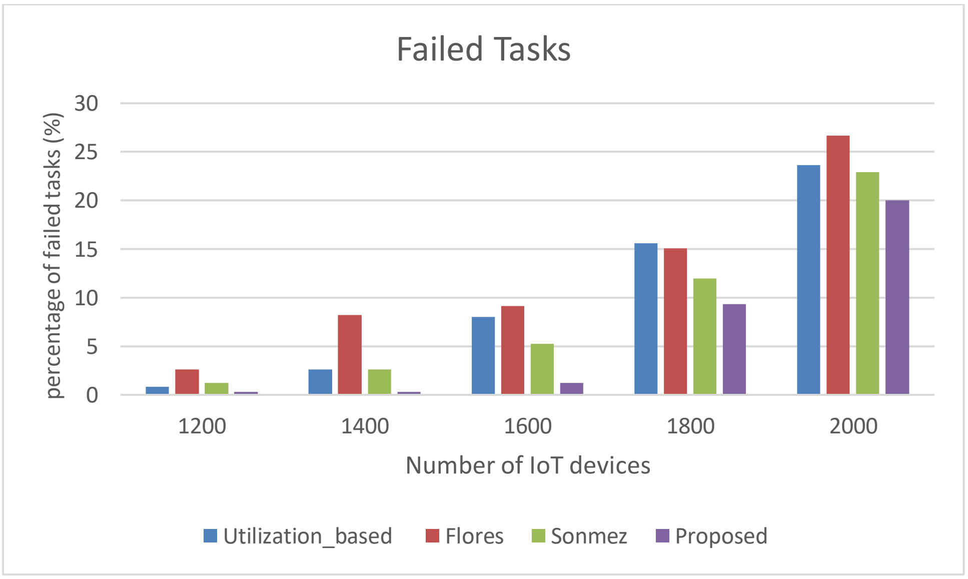


Figure 5. 11: Percentage of failed tasks of the proposed approach with other related approaches Part 2

The results obtained from the preliminary analysis of the average VM utilisation at edge servers are presented in figure 5.12. It represents when the system server IoT devices up to 1000. It can be seen that the utilisation level of al the approaches at 200 devices are similar, and then change when the number of devices increased. The proposed approach is keeping the utilisation level relatively low comparing to other approaches when the number of devices increased. All the other approaches have nearly the same level for all scenarios. On the other hand, as shown in figure 5.13, when the system load is high, the proposed approach was the lowest compared to other algorithms because it trades utilisation for reduced service time. Also, it can be seen that Sonmez and the proposed approach were relatively similar and lower than the others. Flores was the highest and we can link that with results of failed tasks because it assigns the tasks to a highly utilised overloaded VM. Moreover, the proposed approach succeeded to avoid to reach the exponential deterioration when the computational resources reach their limit comparing to other existing approaches. When the resources reach their limit, this will increase the overall service time and the task failure due to insufficient computational resources.

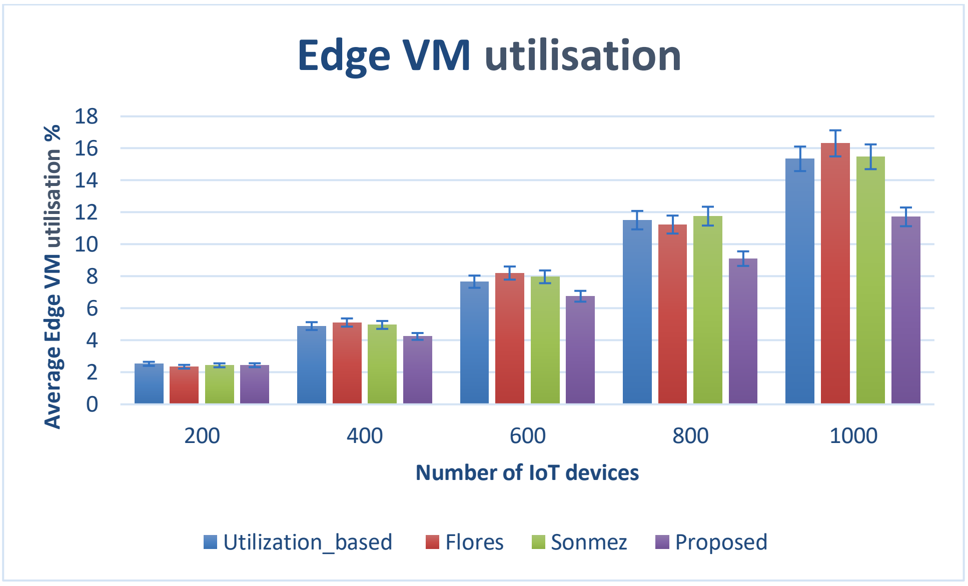


Figure 5. 12: Edge server utilisation of the proposed approach with other related approaches Part 1

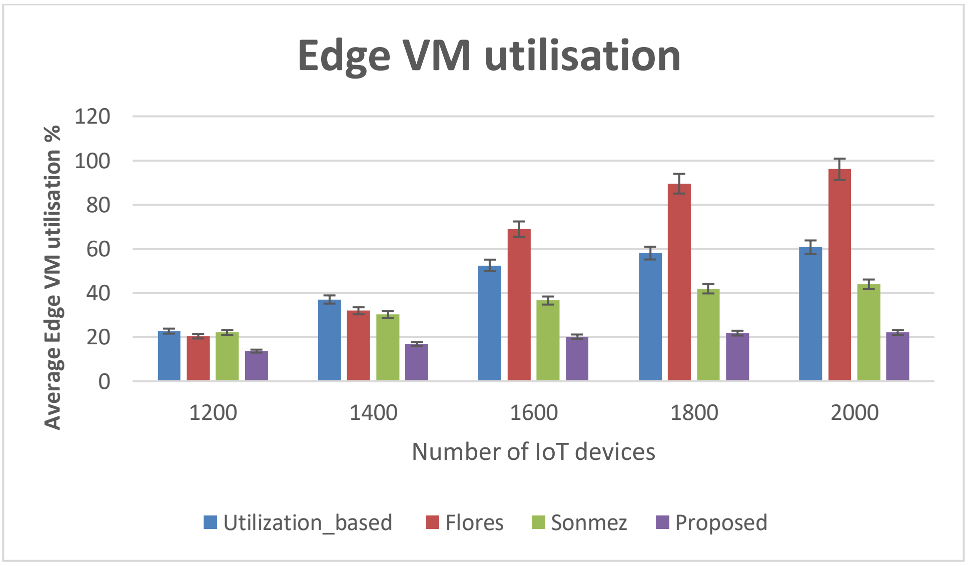


Figure 5. 13: Edge server utilisation of the proposed approach with other related approaches Part 2

5.6 Discussion

The proposed approach was compared against existing related works using a simulation tool EdgeCloudSim, and it was evaluated in the domain of the edge-cloud environment where it was found to improve the overall service time and task failure for latency-sensitive applications as well as effectively utilising the edge-cloud resources.

The obtained results from the simulation tool can be useful and justifiable in the context of the area of this thesis for two reasons:

1- This research aims to evaluate the impact of computational and communication demands of latency-sensitive applications as well as the effectiveness of offloading strategies in order to minimise the overall service time and improve resource utilisation. EdgeCloudSim provides the ability to address this aim. More details are available in Section 2.6.

2- Several published studies [88][144][85][145][146] that have the same interest in the field used EdgeCloudSim to implement and evaluate their works. This lead to increase the trust of the obtained simulation results.

However, EdgeCloudSim has some limitation. According to [147] EdgeCloudSim uses a single server queuing model to calculate the communication delay. This considers as not represented the all available network technologies and could be limited the obtained results. Moreover, It does not support the VM migration between Edge-Cloud resources, which could help to reduce the latency, improve the utilisation and task failures.

On the other hand, there are also a few limitations on the proposed approach. In this research, we assume that we know the required parameters (e.g. task length, the amount of transferred data for uploading and downloading data) in advance which might not always be accurate due to the impact of some other factors. For example, task length is not the only parameters that detriment the CPU time, other parameters such as retrieving data from memory and I/O could affect the CPU time. Additionally, the network time of transferred data affected by other factors such as network congestion. Therefore, methods such as Reinforcement Learning could be useful to measure the effectiveness of the offloading decision by observing each action and train the system to have accurate decisions.

In addition, this work trades utilisation for reduced overall service time; thus, it could lead to wastage in both computation power and resources at the edge level. Other energy efficiency techniques (e.g. VM migration, scaling horizontally/vertically etc.) might be used in the future to overcome this issue and strike a balance between satisfying the demands of applications and utilising the Edge/Cloud computational resources efficiently.

5.7 Summary

This chapter has presented and evaluated a new approach for task offloading in the Edge-Cloud Systems. This approach considering applications characteristics (e.g. CPU, network and delay) as well as the dynamicity of resource utilisation. Moreover, it considers different types of computational resources which represent the real-world scenario. The results show that the proposed approach works effectively with task offloading more than other related approaches in term of overall service time and resource utilisation. It can also reduce the overall task failures due to issues in both network and computational resources.

Chapter 6. Conclusion and Future Work

In this chapter, the summary of the conducted research is presented in section 6.1. The main contributions of the research are followed in section 6.2. Section 6.3 provides an overall evaluation of the research in terms of research objectives in chapter 1. Finally, some future work directions in the area of offloading tasks and resource management of edge computing are presented in section 6.4

6.1 Research Summary

The work in this thesis is focussed on investigating and exploring service delays and the main factors of latency in the edge-cloud system for Latency-sensitive applications. The research is centred on providing an approach to consider all the related parameters from both applications characteristics and the edge-cloud resources in order to minimise the overall service time. The proposed approach is used to minimise the service time of latency-sensitive application and enhance the resource utilisation in the edge cloud system. To the best of our knowledge, this is one of the early attempts to characterize such a system with respect to application's demands (computational and communication), edge-cloud resource utilisation and resource heterogeneity by adopting Fuzzy logic. The results of this study show that the overall service time will never be truly minimised unless the network time is considered in the offloading process. Also, in our findings, the impact of computational demand affected the overall service time more than the communication demand, especially when there is a high increase in the end devices. Moreover, selecting the appropriate resources from a pool of heterogeneous resources could enhance the service time performance and resource utilisation more than double and 40% respectively. In the following a summary of each thesis chapters.

- Chapter 2 presents an overview of the fundamental concepts of the subject of scheduling offloading tasks in the edge-cloud system. It starts with the core concepts of cloud computing with more details on its definition, architecture, deployment models and the idea of mobile cloud computing as an extended model for cloud computing. Then the core concepts of the transformation to the edge computing and its models are introduced. These presented the idea of edge computing and explain the different terms such as fog computing, mobile edge computing, etc., with a comparison between them. Also, the concept of the internet of things (IoT) and its applications are described. After that, the concept of offloading tasks is introduced and discussed with the context of edge computing. This is followed by positioning the work in the related literature, focusing on the scheduling offloading tasks issues and resource management in Edge-Cloud system. A reviewing with related works that focus on application characteristics is presented. Also, the related works that consider parameters of edge cloud resources such as resource utilisation and resource heterogeneity is provided. Finally, research open challenges and simulation tools are presented.
- Chapter 3 presents the overview of the edge-cloud system architecture that supports scheduling offloading tasks of IoT applications, as well as the explanation of the required components and their interactions within

the system architecture. Furthermore, it presents the offloading latency models that consider computation and communication as crucial parameters with respect to offloading to the local edge node, other edge nodes or the cloud. Chapter 3 concludes by discussing early experiments conducted on EdgeCloudSim to investigate and evaluate the latency models of three different offloading scenarios.

- Chapter 4 presents and discusses the main factors of service latency that will be considered in the proposed approach for edge-cloud resource management. Since the demand for computation and communication tasks vary in IoT applications, this chapter aims to validate the impact of these factors on the overall application latency. Moreover, Edge-Cloud environment consists of heterogeneity of computing resources; thus, selecting the appropriate resources to process the offloading tasks play a critical role to improve the overall service time. Therefore, a number of simulation experiments were set up to evaluate the influence of these factors.
- Chapter 5 proposes a new approach for task offloading in edge-cloud systems in order to minimise the overall service time for latency-sensitive applications. The approach adopts the fuzzy logic algorithm that considers application characteristics (i.e. CPU demand, network demand and delay sensitivity) as well as resource utilisation. A number of simulation experiments have been conducted in order to evaluate the proposed approach with other related work.

6.2 Research Contributions

The main contributions of this work can be summarised as follows:

- An investigation on services latency of different tasks' offloading scenarios in the Edge-Cloud environment in order to enhance the end to end service time of latency-sensitive applications. It provides in-depth analyses of the offloading latency models that consider computation and communication as key parameters with respect to offloading to the local edge node, other edge nodes or the cloud. The service latency of the three different offloading scenarios is modelled in order to address the first research question (Q.1). The obtained results in Chapter 3 show that the effectiveness of offloading to the local edge and other edge-cloud resources.
- Quantifying the impact of the variations of the offloading tasks and the performance of different computational resources within the edge-cloud system. The details analysis of the main factors that affect the overall service time is presented in chapter 4 to address research questions (Q.2 and Q.3). Different computation and communication demands of offloading tasks, as well as different VMs, have been modelled in the simulation tool, which has helped to quantify the impact of computation and communication demands of offloading tasks. The results presented in Chapter 4 show that the variation of tasks' demands and VM capabilities have a significant influence on the overall service time.
- Proposed a new approach that adopts the fuzzy logic algorithm to considers application characteristics (e.g., CPU demand, network

demand and delay sensitivity) as well as resource utilisation and resource heterogeneity in order to minimise the overall service time of latency-sensitive applications. Different approaches to schedule offloading tasks are simulated in order to evaluate the proposed approach and to address the research question (Q.4). The obtained results show that the scheduling algorithms of offloading tasks that not considering application characteristics and system behaviour could lead to service time degradation for latency-sensitive applications.

6.3 Overall Research Evaluation

The research objectives of this thesis were discussed In Chapter 1. The section below describes the success of this thesis in achieving these objectives.

- It explored the issues related to scheduling of task offloading in the edge-cloud paradigm. This thesis has reviewed, in Chapter 2, a number of related works that focused on offloading tasks in edge-cloud environments. These have been classified into two main objectives, related works that considering application characteristics and other works that considering resources parameters.
- Investigating the parameters that influence the overall service time in edge-cloud environments. This thesis has presented in Chapter 3 and 4, a number of parameters that has a major effect on the overall service time. The impact of different latency models has been discussed in Chapter 3. Also, the variation of offloading tasks and different computational resources are discussed in Chapter 4.

- Developing a dedicated approach for offloading tasks to handle the requirements of latency-sensitive IoT applications and efficiently utilising the resources in edge-cloud environments in order to minimise the overall service time. Chapter 5 of this thesis has presented the proposed approach for scheduling offloading tasks in order to reduce the overall service time and improve the resource utilisation. It also presented an evaluation with other related approaches through simulation experiments.

6.4 Research Limitations and Future Work

There are a number of future directions with which the work in this thesis could be enhanced. And also, there are several promising directions are related to this work and need to be addressed as highlighted below.

- The proposed approach in chapter 5 considers two types of VMs as the computational resources in the Edge-Cloud environment. Therefore, the approach could be extended to consider more computational resources such as different GPUs and FPGAs since there are many applications for AR/VR and video gaming that requiring intensive computational in order to process their tasks.
- This work handles the scheduling process of independent tasks; however, task dependency plays an essential factor to affect the decision of scheduling tasks. Thus, this work can be extended to consider tasks dependency in the process of scheduling offloading tasks. Tasks dependency and the intercommunication between tasks

can be represented as DAG, which can be modelled within the proposed approach to enhance the overall service time of latency-sensitive applications.

- Another complement work that will enhance the work presented in this thesis is to predict the behaviour of latency-sensitive applications. The prediction can be in several area such as predicting the volume of incoming tasks, predicting the users' mobility which could help to determine their locations. Therefore, it would help the resource manager to prepare the required resources in advance and avoided any performance degradation. This extension would be useful when scheduling offloading tasks in order to minimise the overall service time.
- An extensive and rigorous evaluation of simulation-based results can be done through two main methods. The first method is to implement edge-cloud providers' infrastructure in the real world. To the best of our knowledge, there are only two providers delivering edge services: Amazon lambda and IoT Azure. Unfortunately, measuring all the needed parameters is not allowed on their platforms, but perhaps this will change in the future. The second method is to use two or more simulations to validate the simulation results and compare different tools that use the area. Part of this research can be conducted with more than one simulator tool. For example, the impact of the communication demands and computational demands can be implemented in EdgeCloudSim and IfogSim. However, due to limitations in time, this is not included in this research and can be in future works.

Reinforcement Learning could be a useful method compared to Fuzzy logic to measure the effectiveness of the offloading decision by observing each action and train the system to have accurate decisions. It is a part of machine learning and aims to characterise the learning problem rather than the learning methods. It has been used widely in the area of resource management of Cloud Computing.

List of References

[1] D. Evans, “The internet of things: How the next evolution of the internet is changing everything,” *CISCO white Pap.*, 2011.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013, doi: 10.1016/j.future.2013.01.010.

[3] A. Yousefpour *et al.*, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *J. Syst. Archit.*, vol. 98, no. February, pp. 289–330, 2019, doi: 10.1016/j.sysarc.2019.02.009.

[4] S. Shekhar and A. Gokhale, “Dynamic Resource Management Across Cloud-Edge Resources for Performance-Sensitive Applications,” *Proc. 17th IEEE/ACM Int. Symp. Clust. Cloud Grid Comput.*, pp. 707–710, 2017, doi: 10.1109/CCGRID.2017.120.

[5] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” 2011.

[6] Y. Sahni, J. Cao, S. Zhang, and L. Yang, “Edge Mesh: A New Paradigm to Enable Distributed Intelligence in Internet of Things,” *IEEE Access*, pp. 16441–16458, 2017, doi: 10.1109/ACCESS.2017.2739804.

[7] R. Mahmud, K. Ramamohanarao, and R. Buyya, “Application Management in Fog Computing Environments : A Taxonomy , Review and Future Directions,” vol. 1, no. 1, 2020.

[8] A. Gluhak *et al.*, “A Survey on Facilities for Experimental Internet of Things Research,” *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 58–67,

2014, doi: 10.1109/MCOM.2011.6069710.

[9] A. C. Baktir, C. Sonmez, C. Ersoy, A. Ozgovde, and B. Varghese, “Addressing the Challenges in Federating Edge Resources,” in *Fog and Edge Computing: Principles and Paradigms*, R. Buyya and N. Srirama, Eds. Wiley STM, 2019, pp. 25–49.

[10] G. D’Angelo, S. Ferretti, and V. Ghini, “Modeling the internet of things: A simulation perspective,” *Proc. - 2017 Int. Conf. High Perform. Comput. Simulation, HPCS 2017*, pp. 18–27, 2017, doi: 10.1109/HPCS.2017.13.

[11] C. Sonmez and A. Ozgovde, “EdgeCloudSim : An environment for performance evaluation of edge computing systems,” *Trans. Emerg. Telecommun. Technol.*, no. May, pp. 1–17, 2018, doi: 10.1002/ett.3493.

[12] R. E. Shannon, “Simulation modeling & methodology,” *Proc. - Winter Simul. Conf.*, vol. Part F1308, pp. 9–15, 1976, doi: 10.1145/1102766.1102770.

[13] R. Buyya, R. Ranjan, and R. N. Calheiros, “Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities,” *Proc. 2009 Int. Conf. High Perform. Comput. Simulation, HPCS 2009*, pp. 1–11, 2009, doi: 10.1109/HPCSIM.2009.5192685.

[14] S. Svorobej *et al.*, “Simulating fog and edge computing scenarios: An overview and research challenges,” *Futur. Internet*, vol. 11, no. 3, pp. 1–15, 2019, doi: 10.3390/fi11030055.

[15] A. Azadeh, S. N. Shirkouhi, and K. Rezaie, “A robust decision-making methodology for evaluation and selection of simulation software package,” *Int. J. Adv. Manuf. Technol.*, vol. 47, no. 1–4, pp. 381–393, 2010, doi: 10.1007/s00170-009-2205-6.

[16] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, “The characteristics of cloud computing,” *Proc. Int. Conf. Parallel Process. Work.*, pp. 275–279, 2010, doi: 10.1109/ICPPW.2010.45.

[17] T. Dillon, C. Wu, and E. Chang, “Cloud computing: Issues and challenges,” *Proc. - Int. Conf. Adv. Inf. Netw. Appl. AINA*, pp. 27–33, 2010, doi: 10.1109/AINA.2010.187.

[18] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010, doi: 10.1007/s13174-010-0007-6.

[19] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: architecture, applications, and approaches,” *Wirel. Commun. Mob. Comput.*, pp. 1587–1611, 2013, doi: 10.1002/wcm.

[20] A. N. Khan, M. L. Mat Kiah, S. U. Khan, and S. A. Madani, “Towards secure mobile cloud computing: A survey,” *Futur. Gener. Comput. Syst.*, vol. 29, no. 5, pp. 1278–1299, 2013, doi: 10.1016/j.future.2012.08.003.

[21] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, “Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing,” *J. Netw. Comput. Appl.*, vol. 59, pp. 46–54, 2016, doi: 10.1016/j.jnca.2015.05.016.

[22] D. Kovachev, Y. Cao, and R. Klamma, “Mobile Cloud Computing : A Comparison of Application Models,” *Inf. Syst. J.*, vol. abs/1107.4, no. 4, pp. 14–23, 2011, doi: 10.1007/s11042-012-1100-6.

[23] M. R. Rahimi, J. Ren, C. H. Liu, A. V. Vasilakos, and N. Venkatasubramanian, “Mobile cloud computing: A survey, state of art and future directions,” *Mob. Networks Appl.*, vol. 19, no. 2, pp. 133–143, 2014, doi: 10.1007/s11036-013-0477-4.

[24] Y. Xu and S. Mao, “A survey of mobile cloud computing for rich media applications,” *IEEE Wirel. Commun.*, no. June, pp. 46–53, 2013.

[25] Y. Jararweh, A. Doulat, O. Alqudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, “The future of mobile cloud computing: Integrating cloudlets and Mobile Edge Computing,” *2016 23rd Int. Conf. Telecommun. ICT 2016*, pp. 1–5, 2016, doi: 10.1109/ICT.2016.7500486.

[26] X. He, Z. Ren, C. Shi, and J. Fang, “A Novel Load Balancing Strategy of Software-Defined Cloud / Fog Networking in the Internet of Vehicles,” *China Commun.*, vol. 13, pp. 140–149, 2016, doi: 10.1109/CC.2016.7833468.

[27] M. Firdhous, O. Ghazali, and S. Hassan, “Fog Computing: Will it be the Future of Cloud Computing?,” *Third Int. Conf. Informatics Appl.*, pp. 8–15, 2014, doi: 10.13140/2.1.3216.7684.

[28] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, “Heterogeneity in mobile cloud computing: Taxonomy and open challenges,” *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 369–392, 2014, doi: 10.1109/SURV.2013.050113.00090.

[29] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, “Fog computing may help to save energy in cloud computing,” *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1728–1739, 2016, doi: 10.1109/JSAC.2016.2545559.

[30] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, “Cloudlets: Bringing the cloud to the mobile user,” *MCS’12 - Proc. 3rd ACM Work. Mob. Cloud Comput. Serv.*, pp. 29–35, 2012, doi: 10.1145/2307849.2307858.

[31] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision

and Challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016, doi: 10.1109/JIOT.2016.2579198.

[32] P. G. Lopez *et al.*, "Edge-centric computing: Vision and challenges," *Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015, doi: 10.1145/2831347.2831354.

[33] M. Satyanarayanan, "The Emergence of Edge Computing," *IEEE Comput. Soc.*, no. June, 2017.

[34] R. Want, B. N. Schilit, and S. Jenson, "Enabling the Internet of Things," *IEEE Comput. Soc.*, vol. 48, no. 1, pp. 28–35, 2015, doi: 10.1007/978-3-319-51482-6.

[35] W. Shi and S. Dustdar, "The promise of Edge Computing," *IEEE Comput. Soc.*, no. 0018, 2016.

[36] K. Djemame, R. Kavanagh, and D. Armstrong, "Energy Efficiency Support through Intra-Layer Cloud Stack Adaptation.," in *13th International Conference on Economics of Grids, Clouds, Systems and Services (GECON 2016)*, 2017.

[37] T. Yaofeng, D. Zhenjiang, and Y. Hongzhang, "Key Technologies and Application of Edge Computing," *ZTE Commun.*, vol. 15, no. 1–2, pp. 26–34, 2017, doi: 10.3969/j.

[38] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit from Software-Defined Networking: A Survey, Use Cases, and Future Directions," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017, doi: 10.1109/COMST.2017.2717482.

[39] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Riviere, "On using micro-clouds to deliver the fog," *IEEE Internet Comput.*, vol. 21, no. 2, pp. 8–15, 2017, doi: 10.1109/MIC.2017.35.

[40] S. Wang, K. Chan, R. Urgaonkar, T. He, and K. K. Leung, “Emulation-based study of dynamic service placement in mobile micro-clouds,” *Proc. - IEEE Mil. Commun. Conf. MILCOM*, vol. 2015-Decem, pp. 1046–1051, 2015, doi: 10.1109/MILCOM.2015.7357583.

[41] I. Petri, O. Rana, J. Bignell, and N. Auluck, “Incentivising Resource Sharing in Edge Computing Applications,” *Econ. Grids, Clouds, Syst. Serv. GECON*, vol. 10537, 2017, doi: doi.org/10.1007/978-3-319-68066-8_16.

[42] J. Xu, B. Palanisamy, H. Ludwig, and Q. Wang, “Zenith: Utility-Aware Resource Allocation for Edge Computing,” *Proc. - 2017 IEEE 1st Int. Conf. Edge Comput. EDGE 2017*, pp. 47–54, 2017, doi: 10.1109/IEEE.EDGE.2017.15.

[43] R. Deng, R. Lu, C. Lai, and T. H. Luan, “Towards power consumption-delay tradeoff by workload allocation in cloud-fog computing,” *IEEE Int. Conf. Commun. - Mob. Wirel. Netw. Symp.*, vol. 2015-Septe, pp. 3909–3914, 2015, doi: 10.1109/ICC.2015.7248934.

[44] W. Hu *et al.*, “Quantifying the impact of edge computing on mobile applications,” *Proc. 7th ACM SIGOPS Asia-Pacific Work. Syst. APSys 2016*, 2016, doi: 10.1145/2967360.2967369.

[45] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, “Challenges and Opportunities in Edge Computing,” *IEEE Int. Conf. Smart Cloud*, pp. 20–26, 2016, doi: 10.1109/SmartCloud.2016.18.

[46] K. Dolui and S. K. Datta, “Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing,” *2017 Glob. Internet Things Summit*, pp. 1–6, 2017, doi:

10.1109/GIOTS.2017.8016213.

[47] S. Yi, C. Li, and Q. Li, “A survey of fog computing: Concepts, applications and issues,” *Proc. Int. Symp. Mob. Ad Hoc Netw. Comput.*, vol. 2015-June, pp. 37–42, 2015, doi: 10.1145/2757384.2757397.

[48] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, “The case for VM-based cloudlets in mobile computing,” *IEEE pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009, doi: 10.7748/ns.3.51.55.s67.

[49] A. Bahtovski and M. Gusev, “Cloudlet challenges,” in *24 International Symposium on Intelligent Manufacturing and Automation*, 2014, vol. 69, pp. 704–711, doi: 10.1016/j.proeng.2014.03.045.

[50] A. Sathiaseelan, A. Lertsinsrubtave, A. Jagan, P. Baskaran, and J. Crowcroft, “Cloudrone: Micro clouds in the sky,” *DroNet 2016 - Proc. 2nd Work. Micro Aer. Veh. Networks, Syst. Appl. Civ. Use, co-located with MobiSys 2016*, pp. 41–44, 2016, doi: 10.1145/2935620.2935625.

[51] B. Varghese and R. Buyya, “Next generation cloud computing: New trends and research directions,” *Futur. Gener. Comput. Syst.*, vol. 79, pp. 849–861, 2018, doi: 10.1016/j.future.2017.09.020.

[52] K. Ha *et al.*, “Adaptive VM Handoff Across Cloudlets,” *Tech. Report-CMU-CS-15-113*, no. June, pp. 1–25, 2015.

[53] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A Survey on Mobile Edge Computing: The Communication Perspective,” *IEEE Commun. Surv. Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017, doi: 10.1109/COMST.2017.2745201.

[54] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” *MCC’12 - Proc. 1st ACM Mob. Cloud Comput. Work.*, pp. 13–15, 2012, doi: 10.1145/2342509.2342513.

[55] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, “Fog Computing: A Platform for Internet of Things and Analytics,” in *Big Data and Internet of Things: A Roadmap for Smart Environments*, vol. 546, no. January, Springer, Cham, 2014, pp. 169–186.

[56] T. H. Luan, L. Gao, Z. Li, Y. Xiang, and L. Sun, “Fog Computing: Focusing on Mobile Users at the Edge,” *eprint arXiv:1502.01815*, pp. 1–11, 2015, doi: 10.1016/j.jnca.2015.02.002.

[57] I. Stojmenovic and S. Wen, “The Fog computing paradigm: Scenarios and security issues,” *2014 Fed. Conf. Comput. Sci. Inf. Syst. FedCSIS 2014*, vol. 2, pp. 1–8, 2014, doi: 10.15439/2014F503.

[58] A. Al-fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, “Internet of Things : A Survey on Enabling Technologies, Protocols, and Applications,” *IEEE Commun. Surv. TUTORIALS*, vol. 17, no. 4, pp. 2347–2376, 2015.

[59] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, “Mobile Edge Computing A key technology towards 5G,” *ETSI White Pap.*, no. 11, 2015.

[60] E. Ahmed and M. H. Rehmani, “Mobile Edge Computing: Opportunities, solutions, and challenges,” *Futur. Gener. Comput. Syst.*, vol. 70, pp. 59–63, 2016, doi: 10.1016/j.future.2016.09.015.

[61] R. Mahmud, R. Kotagiri, and R. Buyya, “Fog Computing: A Taxonomy, Survey and Future Directions,” Springer, Singapore, 2018, pp. 103–130.

[62] M. Al-ayyoub, Y. Jararweh, A. Doulat, O. Alqudah, E. Ahmed, and M. Al-ayyoub, “The Future of Mobile Cloud Computing : Integrating Cloudlets and Mobile Edge Computing,” *2016 23rd Int. Conf.*

Telecommun., no. May, pp. 760–764, 2016, doi: 10.1109/ICT.2016.7500486.

[63] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,” *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017, doi: 10.1109/COMST.2017.2682318.

[64] R. Roman, J. Lopez, and M. Mambo, “Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges,” *Futur. Gener. Comput. Syst.*, vol. 78, pp. 680–698, 2018, doi: 10.1016/j.future.2016.11.009.

[65] C. M. Huang, M. S. Chiang, D. T. Dao, W. L. Su, S. Xu, and H. Zhou, “V2V Data Offloading for Cellular Network Based on the Software Defined Network (SDN) Inside Mobile Edge Computing (MEC) Architecture,” *IEEE Access*, vol. 6, pp. 17741–17755, 2018, doi: 10.1109/ACCESS.2018.2820679.

[66] M. Emara, M. C. Filippou, and D. Sabella, “MEC-Assisted End-to-End Latency Evaluations for C-V2X Communications,” *2018 Eur. Conf. Networks Commun. EuCNC 2018*, pp. 157–161, 2018, doi: 10.1109/EuCNC.2018.8442825.

[67] G. I. Klas, “Fog Computing and Mobile Edge Cloud Gain Momentum Open Fog Consortium, ETSI MEC and Cloudlets,” *White Pap.*, pp. 1–14, 2015.

[68] A. Zaslavsky, C. Perera, and D. Georgakopoulos, “Sensing as a Service and Big Data,” *Proc. Int. Conf. Adv. Cloud Comput.*, pp. 21–29, 2012, doi: arXiv:1301.0159.

[69] L. M. Vaquero and L. Rodero-Merino, “Finding your way in the fog:

Towards a comprehensive definition of fog computing," *Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, 2014, doi: 10.1145/2677046.2677052.

[70] K. Rose, S. Eldridge, and C. Lyman, "The internet of things: an overview," *Internet Soc.*, no. October, p. 53, 2015, doi: 10.1017/CBO9781107415324.004.

[71] A. Čolaković and M. Hadžialić, "Internet of Things (IoT): A review of enabling technologies, challenges, and open research issues," *Comput. Networks*, vol. 144, pp. 17–39, 2018, doi: 10.1016/j.comnet.2018.07.017.

[72] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward Computation Offloading in Edge Computing: A Survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019, doi: 10.1109/ACCESS.2019.2938660.

[73] X. Lyu *et al.*, "Selective Offloading in Mobile Edge Computing for the Green Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 54–60, 2018, doi: 10.1109/MNET.2018.1700101.

[74] T. Q. Dinh, S. Member, J. Tang, Q. D. La, T. Q. S. Quek, and S. Member, "Offloading in Mobile Edge Computing : Task Allocation and Computational Frequency Scaling," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3571–3584, 2017.

[75] H. Flores *et al.*, "Large-scale offloading in the Internet of Things," *2017 IEEE Int. Conf. Pervasive Comput. Commun. Work. PerCom Work. 2017*, pp. 479–484, 2017, doi: 10.1109/PERCOMW.2017.7917610.

[76] F. Samie, V. Tsoutsouras, L. Bauer, S. Xydis, and D. Soudris, "Computation Offloading Management and Resource Allocation for Low-power IoT Edge Devices," in *IEEE 3rd World Forum on Internet of*

Things (WF-IoT), 2016, pp. 7–12.

[77] R. Mahmud, K. Ramamohanarao, and R. Buyya, “Latency-aware application module management for fog computing environments,” *ACM Trans. Internet Technol.*, vol. 19, no. 1, 2018, doi: 10.1145/3186592.

[78] W. Tärneberg *et al.*, “Dynamic application placement in the Mobile Cloud Network,” *Futur. Gener. Comput. Syst.*, vol. 70, pp. 163–177, 2017, doi: 10.1016/j.future.2016.06.021.

[79] S. Wang, M. Zafer, and K. K. Leung, “Online Placement of Multi-Component Applications in Edge Computing Environments,” *IEEE Access*, vol. 5, pp. 2514–2533, 2017, doi: 10.1109/ACCESS.2017.2665971.

[80] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, “Hybrid Method for Minimizing Service Delay in Edge Cloud Computing Through VM Migration and Transmission Power Control,” *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, 2017, doi: 10.1109/TC.2016.2620469.

[81] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, “Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, 2016, doi: 10.1109/JIOT.2016.2565516.

[82] D. Zeng, L. Gu, S. Guo, and Z. Cheng, “Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system,” *IEEE Trans.*, vol. 65, no. 12, pp. 3702–3712, 2016.

[83] Q. Fan and N. Ansari, “Application Aware Workload Allocation for Edge Computing-Based IoT,” *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–

2153, 2018, doi: 10.1109/JIOT.2018.2826006.

[84] S. Azizi, “A priority-based service placement policy for Fog-Cloud computing systems,” vol. 7, no. 4, pp. 521–534, 2019.

[85] C. Sonmez, A. Ozgovde, and C. Ersoy, “Fuzzy workload orchestration for edge computing,” *IEEE Trans. Netw. Serv. Manag.*, vol. 16, no. 2, pp. 769–782, 2019, doi: 10.1109/TNSM.2019.2901346.

[86] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, “Cost-effective processing for Delay-sensitive applications in Cloud of Things systems,” *Proc. - 2016 IEEE 15th Int. Symp. Netw. Comput. Appl. NCA 2016*, pp. 162–169, 2016, doi: 10.1109/NCA.2016.7778612.

[87] Y. Li and S. Wang, “An energy-aware edge server placement algorithm in mobile edge computing,” *Proc. - 2018 IEEE Int. Conf. Edge Comput. EDGE 2018 - Part 2018 IEEE World Congr. Serv.*, pp. 66–73, 2018, doi: 10.1109/EDGE.2018.00016.

[88] V. Scoca, A. Aral, I. Brandic, R. De Nicola, and R. B. Uriarte, “Scheduling Latency-Sensitive Applications in Edge Computing,” in *8TH Intrnathonal conference on cloud computing and services science*, 2018.

[89] D. G. Roy, D. De, A. Mukherjee, and R. Buyya, “Application-aware cloudlet selection for computation offloading in multi-cloudlet environment,” *J. Supercomput.*, vol. 73, no. 4, pp. 1672–1690, 2017, doi: 10.1007/s11227-016-1872-y.

[90] M. Taneja and A. Davy, “Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm,” *Proc. IM 2017 - 2017 IFIP/IEEE Int. Symp. Integr. Netw. Serv. Manag.*, pp. 1222–1228, 2017, doi: 10.23919/INM.2017.7987464.

[91] K. Alwasel *et al.*, “IoTSim-SDWAN: A simulation framework for interconnecting distributed datacenters over Software-Defined Wide Area Network (SD-WAN),” *J. Parallel Distrib. Comput.*, vol. 143, pp. 17–35, 2020, doi: 10.1016/j.jpdc.2020.04.006.

[92] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments,” *Softw. - Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017, doi: 10.1002/spe.2509.

[93] T. Qayyum, A. W. Malik, M. A. Khan, O. Khalid, and S. U. Khan, “FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment,” *IEEE Access*, vol. 14, no. 8, pp. 1–1, 2018, doi: 10.1109/ACCESS.2018.2877696.

[94] C. Rodrigo, R. Ranjan, and R. Buyya, “CloudSim: a toolkit formodeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms Rodrigo,” *Softw. - Pract. Exp.*, vol. 39, no. 7, pp. 701–736, 2009, doi: 10.1002/spe.

[95] A. Yousefpour *et al.*, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *J. Syst. Archit.*, vol. 98, no. December 2018, pp. 289–330, 2019, doi: 10.1016/j.sysarc.2019.02.009.

[96] A. Varga, “OMNeT++,” *Model. tools Netw. Simul.*, pp. 35–59, 2010.

[97] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Architecture &amp; Orchestration,” *IEEE Commun. Surv. Tutorials*, vol. PP, no. 99, pp. 1–1, 2017, doi:

10.1109/COMST.2017.2705720.

[98] N. Choi, D. Kim, S. Lee, and Y. Yi, "Fog Operating System for User-Oriented IoT Services : Challenges and Research Directions," *IEEE Commun. Mag.*, no. August, pp. 2–9, 2017.

[99] D. Santoro, D. Zozin, D. Pizzolli, F. De Pellegrini, and S. Cretti, "Foggy: A Platform for Workload Orchestration in a Fog Computing Environment," *Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom*, vol. 2017-Decem, pp. 231–234, 2017, doi: 10.1109/CloudCom.2017.62.

[100] A. Hegyi, H. Flinck, I. Ketyko, P. Kuure, C. Nemes, and L. Pinter, "Application orchestration in mobile edge cloud : Placing of IoT applications to the edge," *IEEE 1st Int. Work. Found. Appl. Self-Systems, FAS-W 2016*, pp. 230–235, 2016, doi: 10.1109/FAS-W.2016.56.

[101] K. Imagane, K. Kanai, J. Katto, T. Tsuda, and H. Nakazato, "Performance evaluations of multimedia service function chaining in edge clouds," *CCNC 2018 - 2018 15th IEEE Annu. Consum. Commun. Netw. Conf.*, vol. 2018-Janua, pp. 1–4, 2018, doi: 10.1109/CCNC.2018.8319249.

[102] A. Carrega, M. Repetto, and A. Zafeiropoulos, "A Middleware for Mobile Edge Computing," *IEEE Cloud Comput.*, vol. 4, no. 4, p. 12, 2017.

[103] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 38–43, 2017, doi: 10.1109/MCOM.2017.1600249CM.

[104] "New services & applications with 5G ultra-reliable low latency communications," *Americas 5G*, vol. 16, no. 0, p. 60, 2018.

[105] R. A. Dziyauddin, D. Niyato, N. C. Luong, M. A. M. Izhar, M. Hadhari,

and S. Daud, “Computation Offloading and Content Caching Delivery in Vehicular Edge Computing: A Survey,” pp. 1–29, 2019.

[106] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, “Quality of Experience (QoE)-aware placement of applications in Fog computing environments,” *J. Parallel Distrib. Comput.*, vol. 132, pp. 190–203, 2019, doi: 10.1016/j.jpdc.2018.03.004.

[107] X. Sun and N. Ansari, “Latency Aware Workload Offloading in the Cloudlet Network,” *IEEE Commun. Lett.*, vol. 21, no. 7, pp. 1481–1484, 2017, doi: 10.1109/LCOMM.2017.2690678.

[108] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, “Scalability and Performance Evaluation of Edge Cloud Systems for Latency Constrained Applications,” in *Third ACM/IEEE Symposium on Edge Computing*, 2018, no. October, doi: 10.1109/SEC.2018.00028.

[109] S. Shekhar, A. Chhokra, A. Bhattacharjee, G. Aupy, and A. Gokhale, “INDICES: Exploiting Edge Resources for Performance-aware Cloud-hosted Services,” *1st IEEE/ACM Int. Conf. Fog Edge Comput. (ICFEC)(to Appear. Madrid, Spain*, 2017, doi: 10.1109/ICFEC.2017.16.

[110] S. Sarkar, S. Chatterjee, and S. Misra, “Assessment of the Suitability of Fog Computing in the Context of Internet of Things,” *IEEE Trans. Cloud Comput.*, vol. 7161, no. c, pp. 1–1, 2015, doi: 10.1109/TCC.2015.2485206.

[111] K. Sasaki, S. Makido, and A. Nakao, “Vehicle Control System for Cooperative Driving Coordinated Multi -Layered Edge Servers,” *Proc. 2018 IEEE 7th Int. Conf. Cloud Networking, CloudNet 2018*, 2018, doi: 10.1109/CloudNet.2018.8549396.

[112] Y. Gao, W. Hu, K. Ha, B. Amos, P. Pillai, and M. Satyanarayanan, “Are

Cloudlets Necessary?," no. October. School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep, 2015.

[113] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, 2009, doi: 10.1109/MPRV.2009.82.

[114] H. Wu, Y. Sun, and K. Wolter, "Energy-Efficient Decision Making for Mobile Cloud Offloading," *IEEE Trans. Cloud Comput.*, vol. 7161, no. c, 2018, doi: 10.1109/TCC.2018.2789446.

[115] C. Li, J. Tang, H. Tang, and Y. Luo, "Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment," *Futur. Gener. Comput. Syst.*, vol. 95, pp. 249–264, 2019, doi: 10.1016/j.future.2019.01.007.

[116] W. Hu *et al.*, "Quantifying the Impact of Edge Computing on Mobile Applications," *Proc. 7th ACM SIGOPS Asia-Pacific Work. Syst. - APSys '16*, pp. 1–8, 2016, doi: 10.1145/2967360.2967369.

[117] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," *Multimed. Tools Appl.*, pp. 24639–24655, 2018, doi: 10.1007/s11042-018-7051-9.

[118] E. Renart, J. Diaz-montes, and M. Parashar, "Data-driven Stream Processing at the Edge," in *IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, 2017, pp. 1–10, doi: 10.1109/ICFEC.2017.18.

[119] A. H. Sodhro, Z. Luo, A. K. Sangaiah, and S. W. Baik, "Mobile edge computing based QoS optimization in medical healthcare applications," *Int. J. Inf. Manage.*, vol. 45, pp. 308–318, 2019, doi: 10.1016/j.ijinfomgt.2018.08.004.

[120] S. Choy, B. Wong, G. Simon, and C. Rosenberg, “A hybrid edge-cloud architecture for reducing on-demand gaming latency,” *Multimed. Syst.*, vol. 20, no. 5, pp. 503–519, 2014, doi: 10.1007/s00530-014-0367-z.

[121] M. Aldossary and K. Djemame, “Performance and energy-based cost prediction of virtual machines auto-scaling in clouds,” *Proc. - 44th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2018*, pp. 502–509, 2018, doi: 10.1109/SEAA.2018.00086.

[122] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, “QoS Prediction for Service Recommendation with Deep Feature Learning in Edge Computing Environment,” *Mob. Networks Appl.*, vol. 25, no. 2, pp. 391–401, 2020, doi: 10.1007/s11036-019-01241-7.

[123] H. Tan, Z. Han, X. Y. Li, and F. C. M. Lau, “Online job dispatching and scheduling in edge-clouds,” in *Proceedings - IEEE INFOCOM*, 2017, doi: 10.1109/INFOCOM.2017.8057116.

[124] R. Beraldi, A. Mtibaa, and H. Alnuweiri, “Cooperative load balancing scheme for edge computing resources,” in *2017 2nd International Conference on Fog and Mobile Edge Computing, FMEC 2017*, 2017, pp. 94–100, doi: 10.1109/FMEC.2017.7946414.

[125] L. Liu, Z. Chang, X. Guo, S. Mao, and T. Ristaniemi, “Multiobjective Optimization for Computation Offloading in Fog Computing,” in *IEEE Symposium on Computers and Communications (ISCC)*, 2017, vol. 5, no. 1, pp. 283–294, doi: 10.1109/JIOT.2017.2780236.

[126] Z. Ou, H. Zhuang, J. K. Nurminen, A. Ylä-Jääski, and P. Hui, “Exploiting hardware heterogeneity within the same instance type of Amazon EC2,” in *4th Workshop on Hot Topics in Cloud Computing, HotCloud 2012*, 2012, pp. 4–8.

[127] T. Tran and D. Pompili, “Joint Task Offloading and Resource Allocation for Multi-Server Mobile-Edge Computing Networks,” *IEEE Trans. Veh. Technol.*, vol. 68, no. 1, pp. 856–868, 2019, doi: 10.1109/ICC.2019.8761239.

[128] D. Zhou, F. Chao, C. M. Lin, L. Yang, M. Shi, and C. Zhou, “Integration of fuzzy CMAC and BELC networks for uncertain nonlinear system control,” *IEEE Int. Conf. Fuzzy Syst.*, 2017, doi: 10.1109/FUZZ-IEEE.2017.8015410.

[129] L. Abdullah, “Fuzzy Multi Criteria Decision Making and its Applications: A Brief Review of Category,” *Procedia - Soc. Behav. Sci.*, vol. 97, no. November 2013, pp. 131–136, 2013, doi: 10.1016/j.sbspro.2013.10.213.

[130] X. Wei, C. Tang, J. Fan, and S. Subramaniam, “Joint Optimization of Energy Consumption and Delay in Cloud-to-Thing Continuum,” *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1–1, 2019, doi: 10.1109/JIOT.2019.2906287.

[131] L. Yang, J. Cao, H. Cheng, and Y. Ji, “Multi-User Computation Partitioning for Latency Sensitive Mobile Cloud Applications,” *IEEE Trans. Comput.*, vol. 64, no. 8, pp. 2253–2266, 2015, doi: 10.1109/TC.2014.2366735.

[132] A. Mukherjee, D. De, and D. Roy, “A Power and Latency Aware Cloudlet Selection Strategy for Multi-Cloudlet Environment,” *IEEE Trans. Cloud Comput.*, vol. 7161, no. c, pp. 1–1, 2016, doi: 10.1109/TCC.2016.2586061.

[133] S. Sharma and H. Saini, “A novel four-tier architecture for delay aware scheduling and load balancing in fog environment,” *Sustain. Comput.*

Informatics Syst., vol. 24, p. 100355, 2019, doi: 10.1016/j.suscom.2019.100355.

[134] X. Xu *et al.*, “Dynamic Resource Allocation for Load Balancing in Fog Environment,” *Wirel. Commun. Mob. Comput.*, vol. 2018, 2018, doi: 10.1155/2018/6421607.

[135] B. Yang, W. K. Chai, G. Pavlou, and K. V. Katsaros, “Seamless Support of Low Latency Mobile Applications with NFV-Enabled Mobile Edge-Cloud,” *Proc. - 2016 5th IEEE Int. Conf. Cloud Networking, CloudNet 2016*, pp. 136–141, 2016, doi: 10.1109/CloudNet.2016.21.

[136] M. Helbig, K. Deb, and A. Engelbrecht, “Key challenges and future directions of dynamic multi-objective optimisation,” *2016 IEEE Congr. Evol. Comput. CEC 2016*, pp. 1256–1261, 2016, doi: 10.1109/CEC.2016.7743931.

[137] A. Ansari and A. A. Bakar, “A Comparative Study of Three Artificial Intelligence Techniques: Genetic Algorithm, Neural Network, and Fuzzy Logic, on Scheduling Problem,” *Proc. - 2014 4th Int. Conf. Artif. Intell. with Appl. Eng. Technol. ICAIET 2014*, pp. 31–36, 2015, doi: 10.1109/ICAIET.2014.15.

[138] X. Kong, C. Lin, Y. Jiang, W. Yan, and X. Chu, “Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction,” *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1068–1077, 2011, doi: 10.1016/j.jnca.2010.06.001.

[139] F. Basic, A. Aral, and I. Brandic, “Fuzzy handoff control in edge offloading,” *Proc. - 2019 IEEE Int. Conf. Fog Comput. ICFC 2019*, pp. 87–96, 2019, doi: 10.1109/ICFC.2019.00020.

[140] L. Ramaswamy, A. Iyengar, and J. Chen, “Cooperative data placement

and replication in edge cache networks," *2006 Int. Conf. Collab. Comput. Networking, Appl. Work. Collab.*, 2006, doi: 10.1109/COLCOM.2006.361850.

[141] L. Mao, Y. Li, G. Peng, X. Xu, and W. Lin, "A multi-resource task scheduling algorithm for energy-performance trade-offs in green clouds," *Sustain. Comput. Informatics Syst.*, vol. 19, no. January, pp. 233–241, 2018, doi: 10.1016/j.suscom.2018.05.003.

[142] H. Flores and S. N. Srirama, "Adaptive code offloading for mobile cloud applications: Exploiting fuzzy sets and evidence-based learning," *MCS 2013 - Proc. 4th ACM Work. Mob. Cloud Comput. Serv.*, pp. 9–16, 2013, doi: 10.1145/2482981.2482984.

[143] C. Sonmez, "Performance Evaluation of Single-Tier and Two-Tier Cloudlet Assisted Applications," *2017 IEEE Int. Conf. Commun. Work. (ICC Work.*, pp. 302–307, 2017, doi: 10.1109/ICCW.2017.7962674.

[144] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, S. U. Khan, and P. Li, "A Double Deep Q-Learning Model for Energy-Efficient Edge Scheduling," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 739–749, 2019, doi: 10.1109/TSC.2018.2867482.

[145] M. D. Hossain *et al.*, "Collaborative Task Offloading for Overloaded Mobile Edge Computing in Small-Cell Networks," *Int. Conf. Inf. Netw.*, vol. 2020-Janua, pp. 717–722, 2020, doi: 10.1109/ICOIN48656.2020.9016452.

[146] A. Kovalenko, R. F. Hussain, O. Semiari, and M. A. Salehi, "Robust resource allocation using edge computing for vehicle to infrastructure (v2i) networks," *2019 IEEE 3rd Int. Conf. Fog Edge Comput. ICFEC 2019 - Proc.*, 2019, doi: 10.1109/CFEC.2019.8733151.

[147] D. Perez Abreu, K. Velasquez, M. Curado, and E. Monteiro, “A comparative analysis of simulators for the Cloud to Fog continuum,” *Simul. Model. Pract. Theory*, vol. 101, no. October 2019, p. 102029, 2020, doi: 10.1016/j.simpat.2019.102029.