A resource management framework for fog computing networks

Presented by:

Mxolisi Mtshali

Supervisor:

Professor Matthew O Adigun

Dept. of Computer Sciences

University of Zululand

Co-Supervisor:

Mr Zaaid Du Toit

Dept. of Telecoms and Media

Council for Scientific and Industrial Research

Submitted to the Department of Computer Sciences at the University of Zululand in partial fulfilment of the academic requirements for a Master of Science degree in Computer Sciences (Computer Sciences)

February 27, 2020

In memory of my late mother Jabulile Florence Mtshali. You left fingerprints of grace in our lives. Your words will forever be remembered.

Qhawekazi Lethu!!!.

Declaration

1. I acknowledge that I have read and understood the University's policies and rules applicable to postgraduate research, and I certify that I have, to the best of my knowledge and belief, complied with their requirements.
2. I declare that this dissertation, save for the supervisory guidance received, is the product of my own work and effort. I have, to the best of my knowledge and belief, acknowledged all sources of information in line with normal academic conventions.
3. I further certify that the presented research is original, and that the material submitted for examination has not been submitted, either in whole or in part, for a degree at this or any other university.
4. I have subjected this document to the University's text-matching and/or similarity-checking procedures and I consider it to be free of any form of plagiarism.

Mxolisi Mtshali
February 27, 2020

Acknowledgments

Above all, I would like to thank God for being a pillar of my strength throughout this journey, without His gracious help it would have been an impossible mission to accomplish. Throughout, while working on this dissertation, I have gained a lot of interesting and valuable experiences and I was not walking alone, therefore I would also like to express appreciation to people who have been with me through thick and thin till I have finished my dissertation.

First, I would like to thank my supervisor Professor Matthew Adigun who has helped me with guidance, supervision, and constructive criticism until the end of this dissertation work, I can't describe how grateful I am to have had you as my supervisor. Thank you so much for your willingness to assist me whenever I cried for help and for your constant faith in me. May God richly bless you in all your endeavours.

Similar profound gratitude to my mentors Mr Sabelo Dlamini and Dr Pragasen Mudali. Thank you so much for your motivation, patience, enthusiasm, and willingness to share your immense knowledge of networking. Your passion and dedication to develop young people are truly humbling. I am forever indebted to you.

I am also hugely appreciative of my manager Zaaid du Toit. Thank you so much for granting me the time I needed to complete this work. Most importantly, I sincerely appreciate your financial support and words of encouragement. You truly are a great leader and I will forever be grateful. To my colleagues (Lusani Mamushiane, Themba Shozi, Mpho Nkosi and Dr Hlabishi Kobo), my family and friends, your support and love kept me going and I am truly grateful.

Abstract

The evolution of the Internet of Things (IoT) has drastically changed how computing devices can be deployed by enabling them to be located anywhere on the cloud-to-things continuum. This has emerged as a solution to most emerging market challenges. These ubiquitous computing devices do not only possess the data-processing capabilities, they also have computing and storage capabilities. Based on these device capabilities, a new paradigm has evolved. This paradigm decentralizes all the centralized cloud capabilities and locates them at the edge of the network. This is popularly known as fog computing. The goal is to avoid deploying the IoT services to the cloud core servers for processing and storage resources, to also mitigate latency, and cost of deployment.

Since the paradigm is relatively new, there are challenges that need to be tackled in order to have a reliable network deployment. The main challenge of this paradigm arises from the relocation of service from the resource-rich physical underlying infrastructure of the cloud core to the Fog layer where there is limited resources and physical infrastructure, because limited infrastructure offers limited capacity in terms of computing, network, processing and storage resources. This means that should a heavy application be executed on these limited FN services, they can be over-consumed, which could lead to network breakdown or failure owing to device shutdown.

As the resources are the main significant part of the network, they must be able to offer services on demand without being exploited. Therefore, in order to optimize the resource management operation efficiently, a method such as resource scheduling must be applied to fog deployments. To address the matter, factors such as scheduling algorithms and framework are considered. In the process of network design and deployment, critical questions arise, such as: How can a resource management framework be used to address the challenges in fog computing; Why do existing resource scheduling mechanisms not respond adequately to resource management challenges of fog computing networks? Which resource-scheduling algorithms can be used to address specific resource management challenges, and what are their relevant achievements? Henceforth, this challenge will be referred to as a resource management problem. This challenge is a multi- objective problem and comprises competing objectives such as service delay, energy consumption and network utilization. Therefore, it affirms that there is no universal solution available.

Abstract

Therefore, this dissertation proposes a resource management framework as a solution to the network-planning problem. The planning covers the optimal placement of tasks with respect to resource consumption optimization and optimal offloading of tasks in the continuum. Addressing this multi-objective optimization problem involves two stages. First, four classical scheduling methods, namely, first come first serve (FCFS), shortest job first (SJF), round robin (RR), and priority-based (PB) have been evaluated. The optimal method progresses from the first stage to the second stage, where its overall performance is evaluated against an unsupervised machine-learning algorithm K-means. The simulation findings show that, as the number of sensor request scales up, the service delay and the energy consumption of the FCFS scheduling method increase linearly; while in the case of K-means, the service delay increases exponentially. The FCFS method yields optimal results in terms of service delay and CPU execution time spent on the task, and also achieved the best trade-off between the competing objectives of service delay and energy consumption, whereas, the K-means clustering method has optimal energy consumption with high service delay resulting in the worst trade-off. The modelling covers most realistic fog computing applications, parameters and constraints, therefore it can be easily deployed in the fog computing landscape while extending the current cloud architecture.

Abstract

Contents

List of Figures

List of Tables

List of Acronyms

IoT	Internet of Things
EC	Edge Computing
FC	Fog Computing
FCN	Fog Computing Networks
FN	Fog Node
CC	Cloud Computing
MEC	Mobile Edge Computing
ETSI	European Telecommunications Standards Institute
API	Application Programming Interface
SDN	Software Defined Networking
WAN	Wide Area Network
5G	Fifth Generation of Mobile Communications
NIST	National Institute of Standard and Technology
EU	End-User
VFC	Vehicular Fog Computing
CPS	Cyber-Physical Systems
DAG	Direct Acyclic Graph

Chapter 1 Introduction

The concept of having application utilities over the internet offering clients rights to control, manage, create, configure and customize applications without having to worry about the actual resource management and maintenance has been known as cloud computing Gu (2000), Pang et al. (2016), Chen et al. (2017). However, because of the evolution of technology, a relatively new concept, called fog computing, has received more attention Luan et al. (2015), Bierzynski et al. (2017).

Therefore, this chapter covers the basic concept of two distinct computing models, the cloud and the fog. Section 1.1 discusses the dawn of the fog computing concept. Section 1.2 outlines the challenges that this dissertation addresses. Section 1.3 presents the goal and objectives pursued during the dissertation compilation. Section 1.4 depicts and describes the research method adopted in achieving section 1.3 objectives. Section 1.5 highlights the significance of the research done. Section 1.6 outlines the contributions of the research thesis to the literature. Finally, section 1.7 summarizes the structure of the dissertation.

1.1 Preamble

Cloud computing is a model comprising networking services that provide infrastructure to cater dynamically for data networking, management, processing and storage Kshetri (2013), Rao et al. (2015). Cloud computing forms a large pool of elastic resources and has acted as a de facto solution for the past decades by being a driving technology behind the IoT technology for a variety of applications (Vargas Vargas, 2016). However, with the evolution of a body of technologies known as cyber- physical systems, associated knowledge areas such as machine learning, big data, and the next industrial revolution are the reasons behind the increase in the number of IoT devices connected to the cloud. Moreover, new application requirements have evolved as derivatives of the time-criticality of associated operations. Arising from the foregoing is the realisation that cloud computing alone is failing to keep up with the complexity and time-sensitivity of IoT applications. This situation has brought about the need for research to investigate issues such as network performance and high consumption of energy and bandwidth. This has resulted in the dawn of fog computing which the industry is currently witnessing.

In architectural terms, fog computing introduces an intermediary layer of micro- servers known as Fog Nodes, FN, designed with the capability to replicate all cloud core server services such as computation, storage, and processing (Dolui and Datta, 2017). However, FNs have lighter physical underlying resource infrastructures than the cloud infrastructure. They also can be virtualized and clustered or distributed between the centralized cloud and the IoT end-user (EU) devices (Vargas Vargas, 2016). The objective of fog computing is to deliver cloud services seamlessly in response to the prevalence of IoT applications; thereby promoting the technological transformation of the business industry. A crucial advantage of fog computing is allowing data management and analytics at the proximity of EU devices; including useful support to real-time IoT applications enabling the business transformation to occur seamlessly. Moreover, more applications, such as video surveillance, augmented reality, gaming, which are widely distributed along the edge of the network, are taking advantage of FC. This is because geographically distributed FNs can be strategically positioned at optimal locations where virtualized central processing units (vCPU), memory and other computing-related services are offered Zhang et al. (2018b)). From the foregoing, it can be inferred that fog computing is not replacing the cloud computing model but complementing it by improving cloud's overall quality of services; in other words, a lighter data load is pushed to centralized FC servers resulting in cost-effectiveness of small-scale networks. Next, we present the specific mission of this study as a research motivation.

1.2 Research Motivation

This dissertation is a study of the challenge faced when using fog computing nodes to address the exponential growth of generated data from the ever-increasing diversified number of edge devices in the IoT application landscape. Locating a Fog Node at the edge of the network is a challenge because of the limited physical underlying infrastructure allocated for computing, network, and storage. This challenge is addressed in a manner that guarantees enhanced fog computing network performance. The study's approach is to investigate the effect of proper network planning before the actual deployment and adoption of resource management frameworks during the deployment (Pham and Huh, 2016) (Hoang and Dang, 2017).

Designing such optimal resource management is an NP-hard problem (Sehgal et al., 2012) (Wang et al., 2017) (Samie, 2018). Hence, this study addresses the resource-constrained challenge by focusing on the design of the resource management framework in the midst of resource constraints. In particular, the focus of the study is on examining the critical details of the way the framework implements the optimization methods to avoid resource over-consumption on the FNs. Therefore, we propose to extend the scope of existing studies by focusing on implementing resource-scheduling methods that have been implemented for cloud on the fog architecture. In order to optimize objective metrics, a multi-objective optimization problem have been explored using two resource management approaches; namely, *linear-optimization* using classical methods and a *non-linear-optimization* using an unsupervised machine learning clustering method.

1.3 Research Questions and Solution Approach

The problem addressed in this study is to extend the scope of existing studies by focusing on implementing task-scheduling methods that have been implemented for cloud on the Fog architecture by answering the main research question:

How can a network planning approach such as framework be used to address the resource management challenges in the fog computing landscape with respect to resource-scheduling mechanisms?

To answer the main question, a fog computing framework was designed following the concepts of architectural layering and logic related to defined frameworks to host tasks in the fog computing landscape. We focused particularly on mission-critical application systems in the context of Cyber Physical Systems. First, the Fog Nodes represented by the framework and application modules were formulated and modelled based on the application DAG model. Second, optimization methods were modelled with the intention to improve the overall performance of the Fog system. Based on the above approach, the main question was broken down into three sub-questions.

Research question 1: Why do existing resource-scheduling methods not respond adequately to resource management challenges in fog computing networks?

Research question 2: Which resource-scheduling algorithms can be used to address specific resource management challenges, and what are their relative performances?

Research question 3: What are the quantitative trade-offs between service delay and energy consumption of competing objectives?

To answer the first question, application use cases were formulated and modelled based on network parameters such as memory, CPU and link communications. Finally, several resource-scheduling methods including FCFS and K-mean were evaluated and tested on different use cases in the fog computing landscape. To answer the second question, each implemented resource-scheduling method was investigated to assess their impact on network performance. The analysis results presented FCFS and K-mean scheduling methods as competitive based on the objective metrics of service delay and energy consumption. FCFS appeared to be optimal for applications with more focus on receive responses to detected application anomalies, while the K-mean method appeared suitable for applications with Fog Nodes that consume energy optimally.

Finally, the issue of trade-off between the FCFS and K-mean resource-scheduling methods was addressed. The FCFS method had a better trade-off than K-mean in achieving the competing objectives of service delay and energy consumption.

1.4 Research Goal

The goal of this study is to design and develop a resource management framework that enables network operators or providers to implement an optimal fog computing Network, with respect to resource scheduling methods with focus on mission-critical applications.

The above goal is achieved through the success of the following objectives:

Research objective 1: Proposed a mathematical modelling method for a resource management framework showing the dynamics of mission-critical application in a fog landscape.

Research objective 2: Investigated the impact of different optimization methods on the performance of the proposed resource management framework.

Research objective 3: Evaluated and selected the best optimization method that can achieve optimal performance of network aspects such as service delay, energy consumption, and network utilization.

Research objective 4: Carried out quantitative trade-off experimentation and analysis of competing multi-objective metrics such as latency and energy consumption.

1.5 Research Contribution and Publications

The contributions of this dissertation are summarized in this section and matched with associated publications:

- **A resource management framework for fog computing networks, FCN was formulated as an experimentally efficient method of optimizing a network in the Fog infrastructure**. Considering the management of the network's main constraints, including the Fog Node parameters such as memory, vCPU, bandwidth, as well as determining the optimal network links for communications and the task modules' interconnection for seamless operations. Since the main objectives considered are service delay and energy consumption, the resource management framework can help network providers and operators to deploy the Fog Nodes efficiently to have an improved EU experience. The following are the publication outcomes relevant to the contribution discussed:

    1. Mxolisi Mtshali, Sabelo Dlamini, Matthew Adigun and Pragasen Mudali, "An Effective Resource Management Framework for Fog Nodes in Fog Computing Networks", in Southern Africa Telecommunication Networks and Applications Conference (SATNAC), South Africa, Cape Town, 2-5 September.
    Year: 2018 | Conference Paper | Publisher: SATNAC | Accepted.

    2. Mxolisi Mtshali, Sabelo Dlamini, Matthew Adigun and Pragasen Mudali ,"Fog computing as an enabler to the Next Generation Industrial Development," in Southern Africa Telecommunication Networks and Applications Conference (SATNAC), South Africa, Cape Town, 2-5 September.
    Year: 2018 | Conference Paper | Publisher: SATNAC | Accepted.

    3. Mxolisi Mtshali, Sabelo Dlamini, Matthew Adigun and Pragasen Mudali, "Edge Computing for Emerging Markets Addressing African Needs," in IST-Africa Week Conference (IST-Africa), Nairobi, Kenya, 8-10 May.
    Year: 2019 | Conference Paper |Publisher: IEEE | Accepted.

- **Multi-objective optimization technique to address the problem of network performance in fog computing networks was explored by evaluating and analysing relevant task- scheduling and offloading methods**. Amongst the first four linear-optimization methods evaluated during stage one, the FCFS performed better. In stage-two, where FCFS was evaluated against K-means. K-means was the best in terms of energy minimization while FCFS was the best in terms of service latency. However, overall FCFS had a better trade-off analysis percentage than K-Means while for the overall network performance, K-means was optimal when the network utilization objective was considered. Next is the publication outcomes that serve as evidence of this contribution.

  The following are the publication outcomes of the contribution discussed:

  1. 1. Mxolisi Mtshali, Sabelo Dlamini, Matthew Adigun, Pragasen Mudali and Hlabishi Kobo, “Multi-Objective Optimization Approach for Task Scheduling in Fog Computing,” in International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), Drakensberg Sun Resort, South Africa, 5-6 August.
     Year: 2019 | Conference Paper | Publisher: IEEE | Accepted
  2. Mxolisi Mtshali, Sabelo Dlamini, Matthew Adigun and Pragasen Mudali, “Fog Computing Integration with the Internet of Things for Emerging market Use Cases: Architecture, Challenges and Future Directions,” in Southern Africa Telecommunication Networks and Applications Conference (SATNAC), South Africa, Durban, 1-4 September.
     Year: 2019 | Conference Paper | Publisher: SATNAC | Accepted.
  3. Mxolisi Mtshali, Sabelo Dlamini, Matthew Adigun and Pragasen Mudali,” K-Means Based on Resource Clustering for Smart Farming Problem in Fog Computing”, Submitted in AFRICON, GIMPA Executive Conference Centre, Accra, Ghana, 24-27 September.
     Year: 2019 | Conference Paper | Publisher: IEEE | Accepted.

1.6 Dissertation Outline

The dissertation structure is as shown in Figure 1.1:

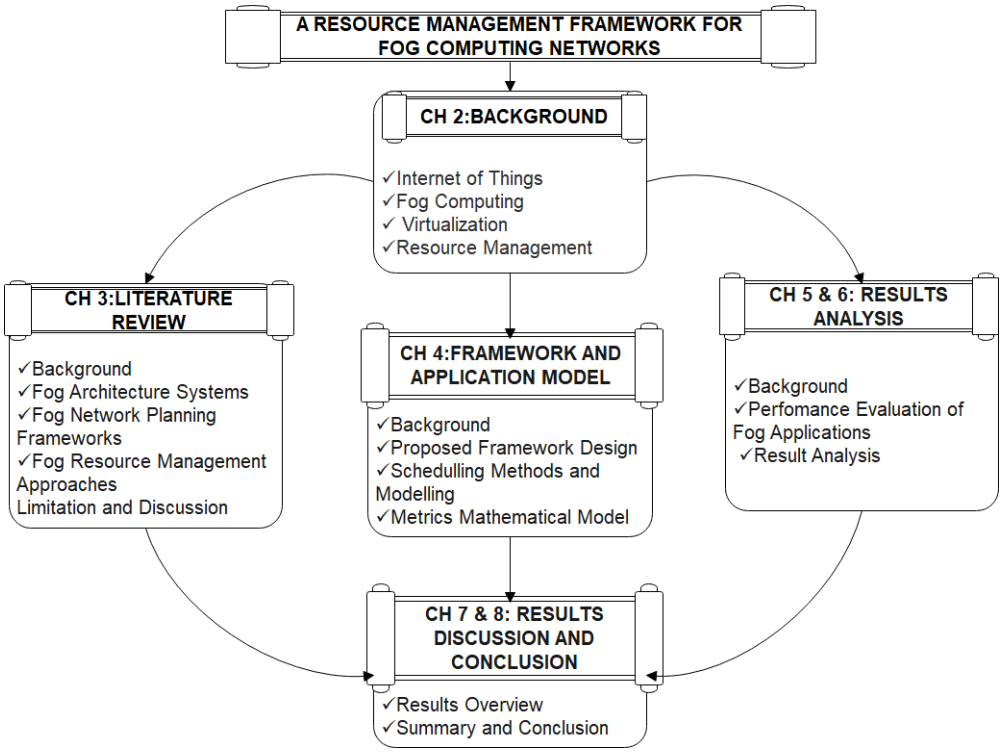

Figure 1.1: Dissertation structure.

In the structure shown in Figure 1.1, we have **Chapter 2** introducing a relevant background review of technologies that form the fundamentals of this dissertation. We begin by discussing the history of technologies such as Internet of Things, fog computing, virtualization and resource management. We especially clarify the concepts and the relations of these technologies. Moreover, we present the distinguishable characteristics of the architecture and service deployments. Finally, we present the concept of resource management in the context of fog computing. In **Chapter 3**, we cover relevant research efforts done in the space of fog computing. In particular, the discussions are based on clarifying several strategies attempted in order to address the efficient design of fog computing systems, the planning of the fog network using frameworks and resource management methods. Finally, we discuss the limitation that needs to be resolved in order to have effective and reliable FCN designs. So, **Chapter 2** and **Chapter 3** address **research question 1**.

Chapter 4 details the functional and non-functional specifications of the requirements for designing a fog computing resource management framework. Moreover, we point out significant assumptions about the framework when resolving the network planning framework in FCNs. We also discuss the selected resource management approaches as forms of management and then present several scheduling methods selected, based on **Chapter 3**. Furthermore, we present the fog-based application designs using mathematical modelling to optimize performance. Finally, in order to implement the method easily, we present the metrics mathematically through formulae. Hence, **Chapter 4** addresses **research question 2**.

Chapter 5 and **Chapter 6** include the actual implementation of the resource management methods modelled in **Chapter 4**. We particularly present two application use-cases, namely ***video surveillance*** and ***Smart farming,*** for examining network performance. Moreover, we have implemented the resource management methods and then evaluated their performance. Two resource management approaches were implemented in ***linear-optimization*** using classical methods and ***non-linear-optimization*** using an unsupervised machine learning clustering method. Finally, we have evaluated the performance and the trade-off analysis of each of the implemented resource management methods by observing their impact on the objective metrics discussed in **Chapter 4**. Therefore, **Chapter 5** and **Chapter 6** address **research question 3**.

Finally, we have **Chapter 7** and **Chapter 8** which conclude, summarize and highlight the contributions by this study and provide an outlook for future work.

Chapter 2

Background Review of Fog-Related Concepts

One of the IoT system's requirements is to be able to operate, even during temporarily unavailable internet connections, and the assumption is that because of the industrial IoT applications, billions of IoT devices are expected to send data to the cloud, which may pose a bandwidth challenge. Because of such cloud computing limitations, analysing and processing data at the edge of the network emerged as a solution, and that concept is referred to as fog computing.

Therefore, this chapter introduces the background knowledge and fundamental technologies that have led to the extension of cloud computing to form the fog computing concept, and presents an overview of technologies that enable the construction of the framework previously discussed in Chapter 1. Since the fog concept is relatively new, section 2.1 discusses the technology that has exposed the need for fog computing. Section 2.2 discusses the conceptual parts of fog computing. Section 2.3 presents the fog computing technical layers and the merits of each layer. In section 2.4, the cloud concept service in the context of fog computing is discussed. In section 2.5, the architectural services are discussed in the fog context while the benefits of fog computing are discussed. Section 2.6 presents the concept of virtualization in the fog context. Section 2.7 discusses the concept of resource management through scheduling. Finally, section 2.8 provides the summary of the chapter.

2.1 Internet of Things

The concept of the IoT began to gain interest from the statement, “Internet of things has the potential to change the world as much as the internet did. Maybe even more so” Asthon (2010). An interesting definition of the concept of the IoT states that, “Internet of things is a network of smart objects interconnected through a communication medium” Kobo et al. (2017). The statement and definition presented clearly justify that the IoT is not only a concept, but also a platform comprising advanced physical devices capable of transmitting and receiving data from all over the world where there is an internet connection. The collection of this raw data simplifies applications by enabling efficient decision making autonomously with the goal of monitoring world processes anywhere anytime Madakam et al. (2015). The IoT domain consists of components such as smart sensors, big data from various applications and smart decision-making methods. These components of the IoT have led to cooperation being attracted to the IoT services to speed up the processes involved within the cooperation domain. Therefore, the IoT continues to pace and mature exponentially.

To date, a huge amount of data forms collected from the numerous data-driven applications are sent to the cloud for storage and processing services Yi et al. (2015a). This huge amount of data is collected by the resource-constrained devices from several application deployments and offloaded to the centralized cloud data servers for advanced computing services. The workflow of this process is as shown in figure 2.1.

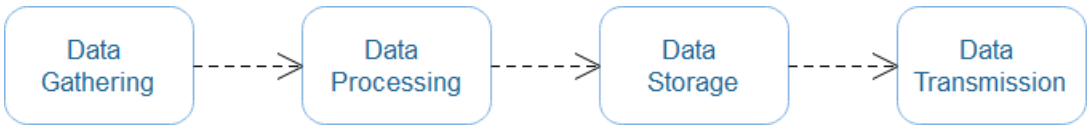

Figure 2.1: The workflow of the IoT application.

In the workflow process, sending data to be cloud-hosted forms the cloud of things (CoT) continuum. The dawn of the CoT continuum arose mainly from the limited computation and storage resources of IoT devices. The cloud is considered a de facto solution model to provide elastic resources to IoT applications. Therefore, the workflow comprises IoT devices for collecting and generating raw data; and finally sending it to the cloud which will provide efficient storage and processing services Yi et al. (2015b). It is therefore apparent that the cloud and IoT capabilities complement each other, although, the cloud has been considered the de facto solution for at least the past decade because once data is located on the cloud there are efficient storage and computing services. However, the prime challenge is to move data to the cloud data servers. Challenges that include bandwidth capacity are an obstruction to the flawless operation of the CoT continuum applications. The fact that the IoT devices produce big data to be fed to the cloud data centres while the network infrastructure cannot handle it all results in network congestion. The network congestion causes increased latency, which fails a new IoT application requirement of low latency.

Moreover, great latency impedes predictability and real-time analytics. This also causes poor quality of services (QoS) to application deliverables. It is such a challenge that has led to the dawn of the fog computing model. The prime objective of fog computing is to solve these latency challenges. Another major challenge that fog computing aims to solve is the generalizability of cloud computing which causes a single point of failure by not being accessible at most geo-distributed locations. This inaccessibility of computing resources could be a problem to mission–critical applications such as in an autonomous production factory.

2.2 Fog Computing

The National Institute of Standard and Technology (NIST) defines fog computing as a "horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional cloud or data centres. This paradigm supports vertically isolated, latency-sensitive applications by providing ubiquitous, scalable, layered, federated, and distributed computing, storage, and network connectivity" Iorga et al. (2017). Fog computing is a model that extend conventional cloud computation services to be located at the edge of the network to mitigate the amount of big data sent to the cloud servers Pham et al. (2017). As previously defined in the introduction, fog computing is not replacing cloud computing; however, it is a subset of cloud computing as fog computing provides edge devices called FNs. The FNs can be any devices that are capable of being high-end servers to provide services such as computation, storage, communication, networking, management, and orchestration. There are several merits that come with decentralizing the processing ability towards the edge of the network, where the big data is generated. Latency is reduced because there are fewer client-server communication hops Kumari et al. (2018). Less bandwidth is required since most big data is now offloaded to the edge for processing, rather than to the core of the cloud Yi et al. (2015c).Geographical distribution of FNs enhances the reliability of applications by enabling fast analytics and better decision making because a single point of failure is eliminated by geo-locating FNs so that each failure can be substituted by nearby available FNs to ensure that redundancy is achieved Brogi et al. (2017).

Fog computing's goal is not to replace the cloud computing model; however, it is an extension that provides big data processing capabilities right at the edge of the network for fast decision-making and analytics. The remaining big data is sent to the cloud for deeper and historical analysis. Fog computing builds up layers of processing between the IoT and the cloud continuum which allows for the interplay of big data between the FNs and the cloud servers Shi et al. (2016).

2.3 Fog Computing Architecture

Fog computing has a vertical hierarchical architecture as illustrated in Fig. 2.2. From this architecture, each layer has the power to handle certain computational requests. The computation can occur at every single level of the pyramid Brogi et al. (2017). In this section, we review the composition of each layer of different fog computing deployments. Fig. 2.2 shows the edge layer comprising FNs that are pushed with raw data from the IoT devices. These FNs focus mainly on filtering, transforming, and compressing the data. After these processes, data is then uploaded to the higher-level FNs. The benefit of the upper-level FNs is the fact that, since they download pre-processed data, they have more overview of the network and also have an improved level of computational intelligence which allows them to make more informed decisions. At the topmost layer, there is the cloud which contains relevant data downloaded from the topmost FNs, and stores it for future analytics and references.

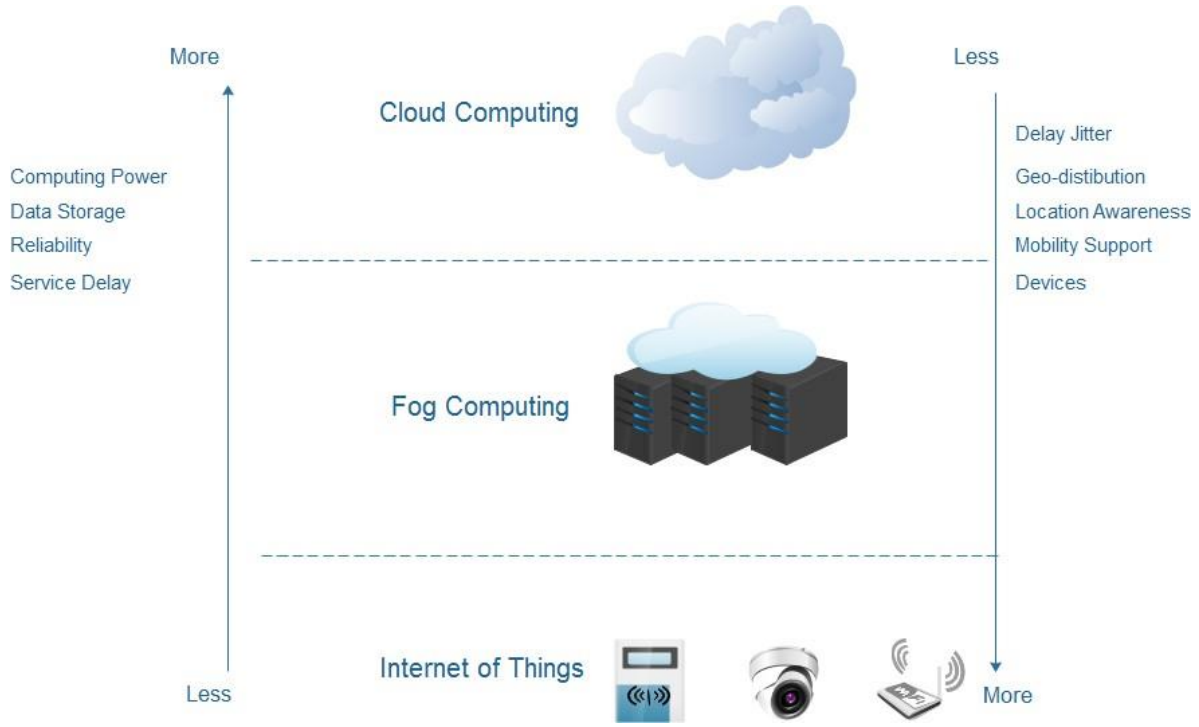

Figure 2.2: High-level fog reference architecture.

The most interesting fact about this type of deployment is the ability of the FNs located at the same level to synchronise to provide robustness, resilience, fault tolerance, and data sharing OpenFog Consortium Architecture Working Group (2017). The peer–to–peer communication is useful for the coordination of handovers.

2.4 Fog Deployment Models

Like the cloud computing model, fog computing comprises public, private, hybrid and community deployment scenarios Dolui and Datta (2017). Since there is no formal concept definition of these different deployments explicitly for fog computing, the cloud concept is adopted. In this section, four deployment concepts and how they can fit into the fog computing domain are compared.

1. *Public fog computing*: here the infrastructure and services are provided offsite across the internet. They offer the elasticity of shared resources. However, they lack security as compared to the private cloud. In the cloud environment, this infrastructure can be owned by one large organization such as Google. However, in fog computing, the members of the public from different organizations can form some collaboration to create adequate infrastructure for all.

2. *Private fog computing*: here the infrastructure and services are maintained exclusively by one business or organization. This private cloud could be hosted either onsite or offsite on the service provider of a third-party organization such as Amazon. This type of cloud can be used in fog computing for healthy and e-healthy applications where data security, privacy, and control are paramount. The private cloud is more flexible, offers improved security and is highly scalable.

3. *Hybrid fog computing*: offers the best of both the public and private clouds. The private cloud maintains the sensitive data, while the public cloud handles the rest of the operations through seamless transition, flexibility, control, and cost-effectiveness.

4. *Community fog computing*: serves organizations with common goals, following the same criteria of security and possibly reliability and sharing of the same resource infrastructure based on their requirements. These organizations share everything, including the cost, and they are more private than public.

2.5 FN Architectural Service Models

Like in the traditional cloud architecture, fog computing has three types of service model implementation, namely software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS).

1. *Software as a service (SaaS)*: This service offers pay per use application software to fog service customers. The applications can run on a cluster of the federated FNs which can be owned by a particular service provider. This service uses the same concept as the traditional cloud paradigm SaaS and therefore this implies that the same methods of application configuration can be applicable to the fog environment. Furthermore, the underlying infrastructure is not managed or controlled by any of the end-users. However, the end users can have their own configurations in the software part that they are renting.

2. *Platform as a Service (PaaS)*: This type of service allows fog customers to deploy applications on the clustered FNs that offer PaaS. In line with the same concept as the cloud, the end-users do not manage or control the underlying infrastructure such as servers, network, storage, and operating systems. However, they can manage the deployed applications and can also adopt their own configuration settings for application-hosting.

3. *Infrastructure as a Service*: This service is similar to the cloud paradigm in offering services such as processing, storage, networks, and other necessary computing resources leveraging the physical infrastructure. However, in the fog paradigm, the FNs have limited computing resources comparable to the cloud. These infrastructure services are offered by FNs that are clustered together and allow customers to deploy software such as operating systems or any other fundamental applications. The customers have access only to control and manage their deployed applications while the IaaS is managed by its providers.

Table 2.1 below presents an overview of the fundamental differences between traditional cloud and fog computing deployment characteristics since they both provide network, computing, and storage capabilities. However, the main aspect of the deployment that evidently contributes to the key differences in characteristics and benefits is architectural behaviour.

Table 2.1: Traditional cloud vs fog computing characteristics.

Characteristics	**Traditional Cloud**	**Fog**
Architecture	Centrally located	Distributed at the network edges Less
Maintenance cost	Higher	Limited Few
Resource Strength	Unlimited	Multiple
Hops	Multiple Few	High
Number of server nodes	Low	High
Real-time analytics Mobility support	Low	

The other advantages of fog computing over cloud are listed and described below:

1. *Scalability*: To reduce a burden of data sent to centralized cloud servers for processing, fog computing provides multiple processing servers distributed at the edge of the network. These endpoints are referred to as FNs and can be extended as the traffic that needs to be handled increases. In addition, FNs can be deployed anywhere, even on a geolocation that has previously been facing difficulties in receiving network coverage.

2. *Privacy*: Data can now be processed locally at the edge of the network instead of being sent to the cloud for analytics. The network management team can have end-to-end control of the entire physical configuration of devices for advanced processes. They can control and keep track of every workflow when collecting and transiting data.

3. *Reliability* : Because the distributed architecture of FNs is located at the edge endpoint of the network, robustness is enabled, which means that if one FN fails, other FNs can avoid network breakdown.

4. *Reduction of Bandwidth*: The FNs at the edge of the network filter all the unnecessary data, because not all raw data can be useful. This process of filtering reduces the amount of data to be sent to the cloud, which allows the cloud to focus only on relevant data. This could be data that will increase the impact required by the system.

5. *Mobility*: It has been troublesome to acquire reliable connectivity for application in mobile devices such as smart driving cars, smart wearable devices, and smartwatches. Fog computing provides portable gateways that enable communication anywhere and everywhere. With this method, fog computing provides an easier way of fulfilling the application demands and thus improves the performance of systems and the quality of services.

2.6 Virtualization vs Containerization in Fog

The concept of virtualization is described as a building block of cloud computing infrastructure which offers abstract hardware resources to multiple customers in an isolated manner. This concept allows for the efficient use of resources while they are shared among multiple customers. Two different types of virtualization are discussed in this dissertation; these are full virtualization and container virtualization and they are shown in Fig. 2.3 Below.

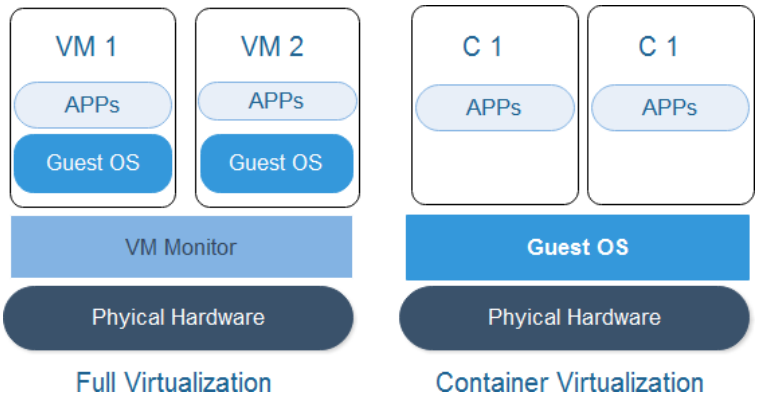

Figure 2.3: Virtualization vs Containerization.

2.6.1 Full Virtualization

Full virtualization is a concept of deploying software applications to run on a physical machine while emulating the features of the hardware system. A layer between the physical hardware and the virtual machine (VM) is known as a hypervisor or a virtual machine monitor (VMM) and it is necessary to virtualize the hardware resources. Multiple VMs can have different operating systems while running on top of the same physical hardware such as a server. Each of these VMs can have a heterogeneous number of CPUs, RAM and other abilities that can be possessed by physical hardware Cao et al. (2018). In complete full virtualization, the workflow process is as follows: The hypervisor which is preventing direct access to the underlying physical hardware is responsible for the hardware management, monitoring, and deployment of VMs on top of it. Each of the deployed VMs can have a homogeneous or heterogeneous operating system running applications on top of it. The benefit of this type of virtualization is the fact that it allows the flexible deployment of heterogeneous hardware resources on top of the same physical hardware resource pool. Although this seems interesting, it also poses challenges such as long starting times caused by the necessity of separating operating systems on each and every VM.

In most cases, VMs are the best for running applications that involve all the operating system's resources and functionality when you need to run multiple applications on servers or have a wide variety of operating systems to manage

2.6.2 Container Virtualization

In contrast to full the virtualization concept, there is the light-weight virtualization known as container virtualization. Container virtualization is a software application running as a process deployed on an operating system and has its own address space while using the same kernel and the physical resources as shown in Fig. 2.3 by Zhang et al. (2018b). In container virtualization, a single physical server can host multiple workloads with a single operating system. In comparison to VMs, the containers can take a few seconds. In addition to the containers, you can have a portable environment such as a Docker container for development, testing, and deployment Yin et al. (2018). In most cases, the containers are beneficiary once prioritizing is maximizing the number of applications running on a minimal number of servers. In the fog computing environment, multiple IoT devices deployed to different geo-locations will send multiple requests at the same time on different federated clusters of FNs. This means each FN will have to run applications while managing the physical resources available on each FN. The multi-tenancy approach should not over-consume the number of physical resources that are virtualized and allocated to each fog service client. Therefore, this dissertation will work on the full virtualization concept for experimentation and design analysis.

2.7 Resource management

This concept is described as an act of delegating storage, networking, computing, and (indirectly) processing power resources to the task application request Sci (2018). The objective is to mutually meet performance goals of applications, infrastructure (e.g., data centre) providers, and the clients. Management can be enabled on either of the above two technologies, namely virtualization and containerization. In the section above, it has been mentioned that this dissertation will focus on full virtualization. However, to manage the resources available on the FNs, methods such as resource scheduling are adopted in this dissertation. This approach to computing management has been proven to optimize the overall network performance, if well applied Dong et al. (2018).

2.8 Chapter Summary

This chapter begins by introducing the concept of the IoT as a technology that has exposed the need for fog computing. Moreover, it discusses the standard concept of fog computing and explores the architecture, technical constraints, deployment models and type of service that can be offered by fog computing. In comparison to the traditional cloud, the fog computing and distributed architecture enhances network performance and application predictability. Additionally, various advantageous over cloud computing are presented. Importantly, the concept of virtualization which makes it easier to optimize services is also explained. Finally, the concept of resource management that facilitates the management of the infrastructure resources is discussed. The research focus is narrowed down in Chapter 3 which presents related work done by other researchers focusing on the fog challenges in the context of the problem introduced in Chapter 1.

Chapter 3

Review of Fog Resource Management Approaches

Fog computing extends the centralized cloud architecture to bring it closer to the edge of the network, which enables efficiency in application, deployment and service offering. In contrast to the cloud, the fog computing architecture is distributed as discussed in Chapter 2. This architecture enables applications to have predictable delays, geo-distribution, and mobility support. Because of such facilities promised by fog, most current and ongoing research focuses on understanding the components of the architecture, network planning and methods to allocate tasks to FNs to improve network performance.

Therefore, this chapter reviews research conducted in the context of fog computing that has been trying to resolve the current challenges in the research field, including what has been highlighted in Chapter 1. Section 3.1 reviews the architectural design requirements for successful network deployment in fog computing. Section 3.2 reviews work done by other research into frameworks design in the context of fog computing. Section 3.3 reviews resource management methods that can critically optimize the performance of the network while avoiding resource over-consumption. Section 3.4 outlines the gaps in other research efforts and presents the summary of the review taxonomy. Finally, Section 3.5. Summarizes the chapter.

3.1 Fog Computing Architecture and Concepts

The ability of fog architecture to scale and distribute the workload in both north/south and east/west flows makes it attractive to the new IoT data dimensions and has attracted researchers to investigate it. In brief, the articles of Bonomi et al. (2012), Shi et al. (2016), Brogi et al. (2017), Zhang et al. (2018a), Zhang et al. (2017) and Kumari et al. (2018) discussed the concept of the fog computing model for the IoT in 2012. In their work, they outlined the challenges that had led to the dawn of fog computing, and they also detail the significance of the paradigm. The main contribution of the fog computing paradigm, according to the authors, was to help the IoT applications to account for requirements such as (i) low latency and context awareness, (ii) wide-spread geographical distribution, (iii) mobility support, (iv) device and network access heterogeneity, and (v) large-scale sensor networks. In order to meet such requirements, an architecture comprising the IoT layer, fog layer, and the cloud layer was presented. According to the authors, their proposed architecture can be suitable for at least three selected applications and other related applications. First, the applications selected include connected cars intercommunicating with smart traffic lights for efficient and flawless mobilizing. Secondly, the smart grids are a smart power supply widely distributed geographically near suppliers and consumers. The third and last use case application is a wireless sensor and actuator network comprising sensor nodes geographically deployed and widely distributed closer to end-user locations. The above-mentioned application use cases have a common main requirement which is real-time processing of the data collected. Furthermore, these applications must be fault tolerant and consume as little energy as possible. To achieve these objectives, a sophisticated fog computing solution is key. Furthermore, authors in (Rao et al., 2015) unpack both the functional and technical components of the fog architecture and also detail the software components of the paradigm. Each layer of the software component of the three-tier architecture presented, is explained in detail. The software components of the architecture are the abstraction layer, fog service and orchestration, and finally, the north-bound application interfaces (APIs). The abstraction layer decouples the heterogeneity of the physical infrastructure over a diversity of platforms providing uniform accessibility to control and management of resources. Secondly, the fog service and orchestration layers are responsible for the policies used during the execution, scheduling and deployment and destruction of FNs. For the communication between the software components of the fog computing architecture, the APIs are utilized. An interesting survey article done by the authors in (Mukherjee et al., 2018) provides more details of the fog computing architectures, concepts and fundamental characteristics of the network. In this article, the authors provide an overview of the service and resource scheduling approaches attempted to address numerous critical issues including delay, bandwidth, and energy-consumption in fog land-scape. Afterward, the survey provides a general overview of

the current state-of-the-art in fog network applications and the design of major research in the aspects of designing fog networks. Finally, highlighted some ongoing research trends, efforts, and open challenges in fog computing.

3.2 FCN Planning Frameworks

The issues related to where/how to deploy applications in the fog landscape are known as a fog network planning problem which has been resolved by making use of the framework concept Yousefpour et al. (2017). Therefore, this section reviews related work that is currently attempting to address the network-planning issues. The authors in (Santos et al., 2017) put forward an autonomous fog computing framework for management and orchestration of smart city applications. This framework aims to provide autonomous resource-provisioning capabilities to fog-computing application management and orchestration. The approach uses an autonomous FN to self-manage during data collection and data analysis. This facilitates quick decision-making functionalities that help to improve the overall network performance. They use the open shortest path first (OSPF) routing protocol to exchange communication between FNs and the fog-cloud communication. This framework mainly aims to assist the application of smart city to be able to detect anomalies and make quick decisions based on them. According to the authors, the framework is able to achieve a substantial reduction in network performance metric such as bandwidth usage compared to the traditional cloud. In addition, the framework can also be used to send timely requests to IoT sensors if an abnormality is detected. The authors advise to look at simulation in future to foresee the challenges and prototypes as proof of concepts of the proposed framework. In (Wang et al., 2017) the authors propose a framework to manage the FN resources. They focus on provisioning and auto-scaling of FN in terms of the amount each node can take before it shuts down. The method used to manage the resources is known as handshaking, where the framework is responsible for deployment and termination of the FN service if anomalies are detected. Furthermore, the auto- scaling method is implemented based on a linear algorithm to manage the resource-constrained FNs. According to the authors, the framework's performance has been investigated with an online game use case known as PokeMon. The experimental results have shown that there is an improvement in the quality of services (QoS) for many end users. Furthermore, the authors report that the results show that there is a 20-80 percent latency improvement when compared to the cloud. The authors also make the point that the amount of latency has been improved by at least 90 percent compared to that of the cloud. Another article in (Santoro et al., 2017) proposes and describes an architectural framework and implements a software platform for resource negotiation and workload orchestration in a highly distributed fog environment. This framework is called Foggy and has been evaluated by using three use cases in video surveillance and vehicle-tracking contexts. The performance analysis is based on metrics such as (i) infrastructure efficiency; (ii) management of resources such

as RAM, CPU and disk in complex scenarios for multi-tenant application and (iii) the method of using scheduling policies related to first come first served (FCFS) to improve the pricing and billing costs. According to authors, results in the framework show the efficiency and optimize the use of infrastructure resource. Article (Zhu et al., 2019) proposes a framework called Folo for improved quality of task allocation and service delay in the vehicular fog computing (VFC) application context. The focus is on mobility support to VFC applications using binary particle swarm optimization (BPSO) and Linear Programming-based Optimization (LBO) to assess the effectiveness of the Folo framework. After simulations, the Folo framework mitigates 56 % of quality loss and 27% of service delay. However, BPSO and LBO are not the only methods that can be applied in the resource scheduling context in the fog computing landscape. Therefore, there is a need to evaluate alternative methods.

3.3 Resource management in FCNs

In order to improve the efficiency of the frameworks discussed in section 3.2, researchers allocate, schedule and offload application loads to available resources using algorithms, and this approach is known as resource management. Resource management is an extensively researched topic in various fields of cloud computing and edge computing as presented in Thomson (2011), Agarwal et al. (2014), Singh et al. (2014) and Chen et al. (2017). However, the challenge is that even though fog computing extends the cloud, the resource management solutions cannot be directly adopted for use in the fog computing environment because the fog comprises distributed architecture that is more dynamic and presents new characteristic. Furthermore, most of the work currently done in cloud computing does not address resource maximization in the fog computing environment with the objective to achieve efficient utilization of network performance features such as latency, bandwidth, energy, and cost of deployment. However, cloud ideas can help achieve the best approaches to demonstrate the resource management method in the fog computing landscape. The research article (Jana and Banerjee, 2018) proposes a resource management technique in the fog computing landscape. The technique is evaluated by using a simulation toolkit known as iFogSim. The authors evaluate performance based on metrics such as execution time, network usage, and energy consumption, as the QoS parameters. The authors' work offers a conclusive decision regarding the results and the method seems to perform better than when it is deployed on the cloud. In addition, the authors concur that the resource management challenge is relatively new and still an open issue that needs to be exhausted for better FCN designs. Another work focusing on fog landscape resource management has been done by (Vilela et al., 2018) who have used a hospital ward as a case study to evaluate and analyse the performance of the simulated scenario in the fog landscape. Their network performance parameters include latency, network usage, cost of transmission and power consumption. The results point out the possibility of enhancing the QoS by deploying

resource management methods at the fog layer. The work presented in (Bittencourt et al., 2017) proposes to use the traditional cloud-scheduling methods in the fog distributed landscape. The goal is to create a dynamic mobility scenario that can evolve dynamically in line with the mobile user's demands. This strategy takes advantage of fog servers' proximity to mobile users and the cloud computing elasticity. In the results report, the fog deployment is shown to be much better than deploying the mobility application in the cloud. It is also clear that implementing the traditional scheduling methods at the fog layer is optimal and can improve the overall performance and QoS of real-time applications. Furthermore, the authors also outline the importance of reviewing and implementing methods for resource management in the fog computing landscape as they are the critical components of network performance. Moreover, (Natesha and Guddeti, 2018) propose the first-fit decreasing (FFD) heuristic approach to solve the challenge of selecting an FN suitable to host the IoT applications. The solution to this placement problem is aimed at reducing the application latency and energy consumption for efficient usage of the FN resource. The validation of the algorithm is evaluated with two placement approaches; either application is deployed in fog-cloud interplay or in cloud only. However, this method has been tested only to be efficient on fog vs cloud. However, it can be extended to consider performance evaluation of scheduling algorithms in fog-only deployment. Furthermore, the work done by (Kabirzadeh et al., 2018) presents various current scheduling algorithms and proposes a hyper-heuristic algorithm for evaluation in fog computing. These scheduling algorithms include, FCFS (First Come First Served), PSO (Particle Swarm Optimization), ACO (Ant Colony Optimization Algorithm), SA (Simulated Annealing Algorithm) and GA (Genetic Algorithm). However, not all algorithms which are alternatives of FCFS have been evaluated, especially when we work on having a cluster of FNs attending to various task requests. To the best of our knowledge, classical methods such as round robin (RR) have only been applied in the traditional centralized architecture Tani et al. (2018), Khurma et al. (2018) . Moreover, work by (Mai et al., 2018) proposes and uses a linear programming method to mitigate the total computational resources of FNs by following a resource-scheduling approach. The outcome of experiments was mitigation of service delay by approximately 16.1% when evaluated against other existing methods. Therefore, this dissertation considers a multi-objective optimization problem study by exploring two resource management approaches namely, *linear-optimization* using classical methods and a *non-linear-optimization* using an unsupervised machine learning clustering methods that have been applied in centralized cloud architectures to the distributed fog landscape for resource management as a method of network performance optimization.

3.4 Limitations and Discussion Summary

In this chapter, we have reviewed a few technological concepts that are fundamental to the research presented by this dissertation. These articles are categorized, compared and analysed appropriately to achieve the goal of this dissertation. The comparison includes a different aspect of technology and authors tackling common conceptual technology. The classification criteria of related work focus more on the following technological concepts.

1. *Internet of Things (IoT)*: These articles review technology that forms the basis of this research and is incorporated in everyday life. Hence, this is needs to be connected to the internet for services such as processing and computation.
2. *Fog Computing Architecture and Concept (FC)*: These articles interpret the current status and help the researchers to have a clear view of the state of fog structure and standard techniques for designing the FCNs.
3. *FCN Planning Frameworks (FCNPF)*: The articles discussed in this section include methods currently used to enable the dynamic adaptation of systems during run-time. The topologies are mostly distributed in the structure.
4. *Resource Management (RM)*: The articles explored in this section use programming models for implementing methods in a distributed architecture to enhance comparable characteristics to achieve an advanced fog computing framework that can be utilized to provide report analyses of aspects of network performances such as latency, network utilization, execution time, energy consumption and cost of execution.

The above-selected concepts are discussed according to their relevance for the work presented by this dissertation, which is a resource management framework for FCNs. The discussion starts with motivation for the work which is the IoT that plays a major role in the development of the framework. Hence, considering the IoT in any discussion of related work is very critical. The second concept is the dynamicity of the designed topology in the fog computing landscape. The dynamicity is discussed and considered because the framework needs to be flexible for diverse applications with the same requirements. According to the reviewed research articles, the current frameworks are based on using resource management methods and improved mathematical techniques. Hence, in the work related to this dissertation, we have considered the resource management techniques and mathematical methods. Fig. 3.1 is the taxonomy of the main studies forming the basis of the work related to this dissertation study and have been unpack on this section above.

Fig. 3.1 classifies the existing solutions to the fog resource management problem.

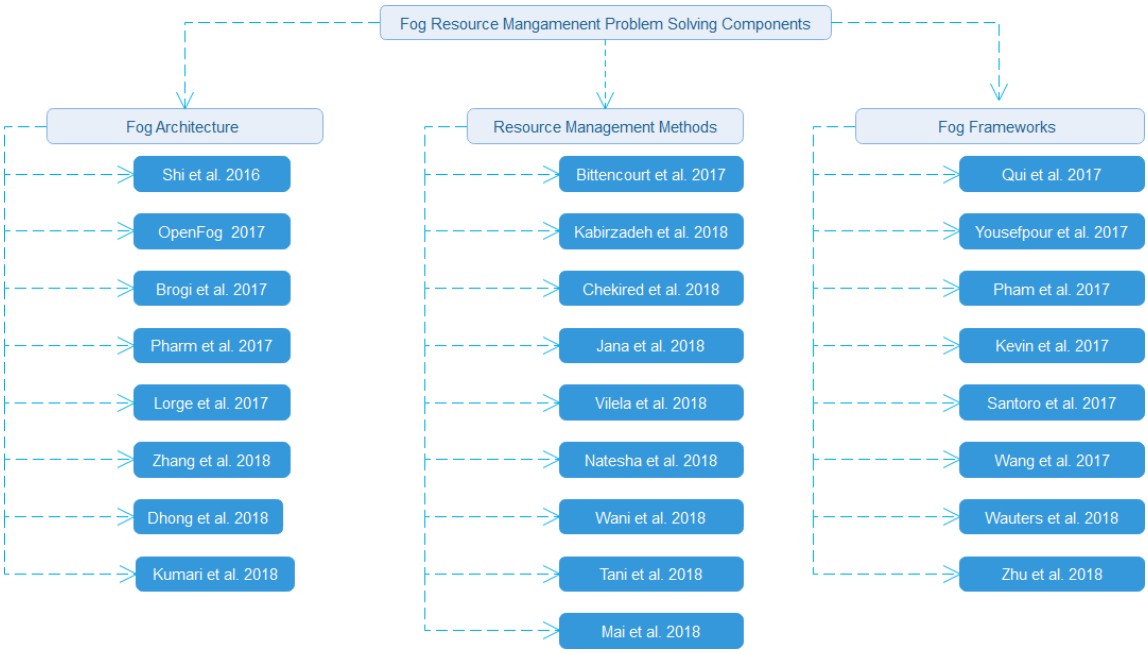

Figure 3.1: Taxonomy of related work

3.5 Chapter Summary

According to the literature review presented in this dissertation, fog computing is an important concept, especially for IoT applications. Fog computing complements the cloud landscape for better service rendering to end users. As a developing technology, fog is still exposed to several challenges that attract researchers and have to be addressed before. Most significantly, proper design of resource management mechanisms needs to be adopted in order to have a network that is fair to both service providers and end users of the fog services.

To the best of our knowledge, this work is the first of its kind to deal with designing a fog resource management framework and extending the scope of existing studies by focusing on energy consumption, network utilization and service delay as objective metrics, and applying resource management techniques to optimize the overall performance in the fog landscape. Consequently, Chapter 4 presents the mathematical methods used in the design of the fog resource management framework.

Chapter 4

Fog Computing Framework and Application Modelling

Since the fog computing model is relatively new, it presents research gaps in the space of fog network planning (Zhang et al., 2019), resource management (Samie, 2018), (Sehgal et al., 2012), (Jana and Banerjee, 2018) and scheduling methods (Yin et al., 2018) as discussed in Chapter 1. In most cases, finding solutions to such challenges results in a multi-objective optimization problem, where more than one classical scheduling methods needs to be bench-marked to optimize performance and trade-offs between competing objectives Liu et al. (2018a), Liu et al. (2018b). To resolve these multi-objective optimization issues, this chapter covers the architectural components and infrastructure design of the planned framework. In brief, communicating entities are unpacked and depicted. Then, inspired by resource scheduling as an approach to the resource management, this chapter proposes the application modelling of four classical scheduling methods using linear optimization approach and then proposes to use an unsupervised machine-learning clustering method using a non-linear-optimization approach to cluster task modules relevant to resource-efficient FNs.

Therefore, this chapter comprises the following sections: Section 4.1 outlines and discusses the dynamicity of the framework components and presents the task workflows. Section 4.2 reviews the partitioning of the application tasks and attributes as modelled in the iFogSim toolkit. Section 4.3 unpacks the scheduling approaches and then models them. Section 4.4 defines the focus of the metrics. Finally, section 4.5 summarizes the chapter.

4.1 Overview of Framework

In almost every network deployment, there is an essential guidance structure supports the basic concepts of the expected functionality, and that is known as the network-planning framework Xiao and Krunz (2017), Pham et al. (2018). Therefore, the framework in Fig. 4.1 is aimed at estimating the effect of task loads sent to the FNs by sensors in order to detect impacts and apply certain scheduling methods to optimize network performance where possible.

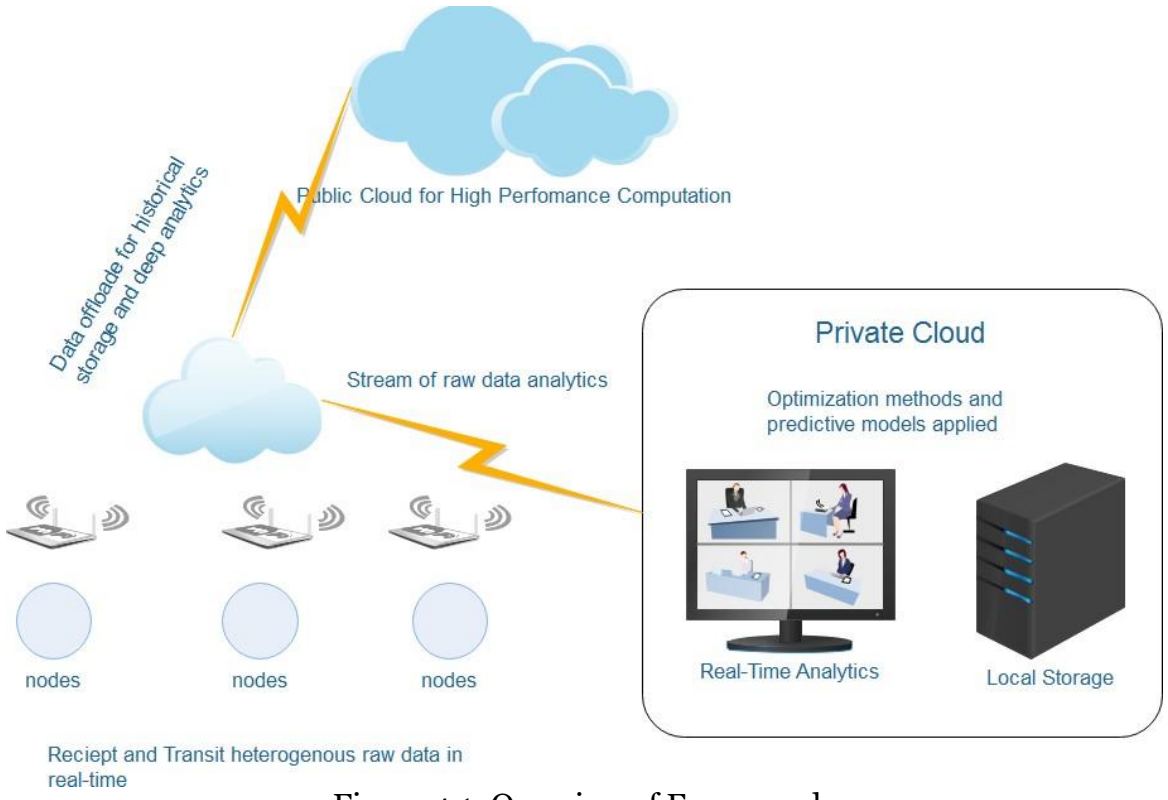

Figure 4.1: Overview of Framework.

Fig. 4.1 attempts to address the resource management problem in fog networks. Initial nodes, which in this case refer to sensors deployed in the area of interest, are integrated with machines to collect information about the current state. The details of the current state are collected as raw data and loaded to the upper level of FNs through wireless communication. Since FNs have limited capacity, they need to calculate in advance the load of the applications they can host. This calls for predictive models to estimate the amount of resources available and then use optimization methods to influence the application scheduling positively as a facet of the resource management mechanism. The analytics are provided in real-time, thus making the framework efficient for mission critical applications.

Moreover, the FN can host multiple nodes in its local storage, and can also provide service, even when connectivity is temporarily unavailable. Where the raw data have been analysed, optimized and then stored, it can now be offloaded to the cloud landscape for historical or further analysis. Offloading simply means that the resources of the FNs are now freed which enables more smooth service to applications. In addition, localising the FNs allows for fast return times and therefore promises service delays to be cleared quickly. In the framework presented the components of the communication are pushed with appropriate information that complies with the modelling and setup presented in this chapter.

The framework focuses on solving the key constrains as a multi-objective optimization problem consisting of competing objectives. For the multi-objective problem to measure the performance of competing objectives, four linear-optimization methods namely First-Come-First-Served (FCFS), Shortest-Job-First (SJF), Round-Robin (RR), and Generalized-Priority (GP) are evaluated by applying the application modelling presented in this chapter. More precisely, the results are analysed and the best-performing method is selected and applied in the second phase. However, in the second phase, a new unsupervised machine-learning (ML) classification technique popularly known as K-Means is applied in the fog network and the performance is examined.

Finally, the results of the K-Means are evaluated against the performance of the optimal method from phase 1. Then the results are reviewed and unpacked to gain better understanding. Fig. 4.2 below presents the workflow approach to solving the resource management problem by using both methods, linear-optimization and non-linear optimization method.

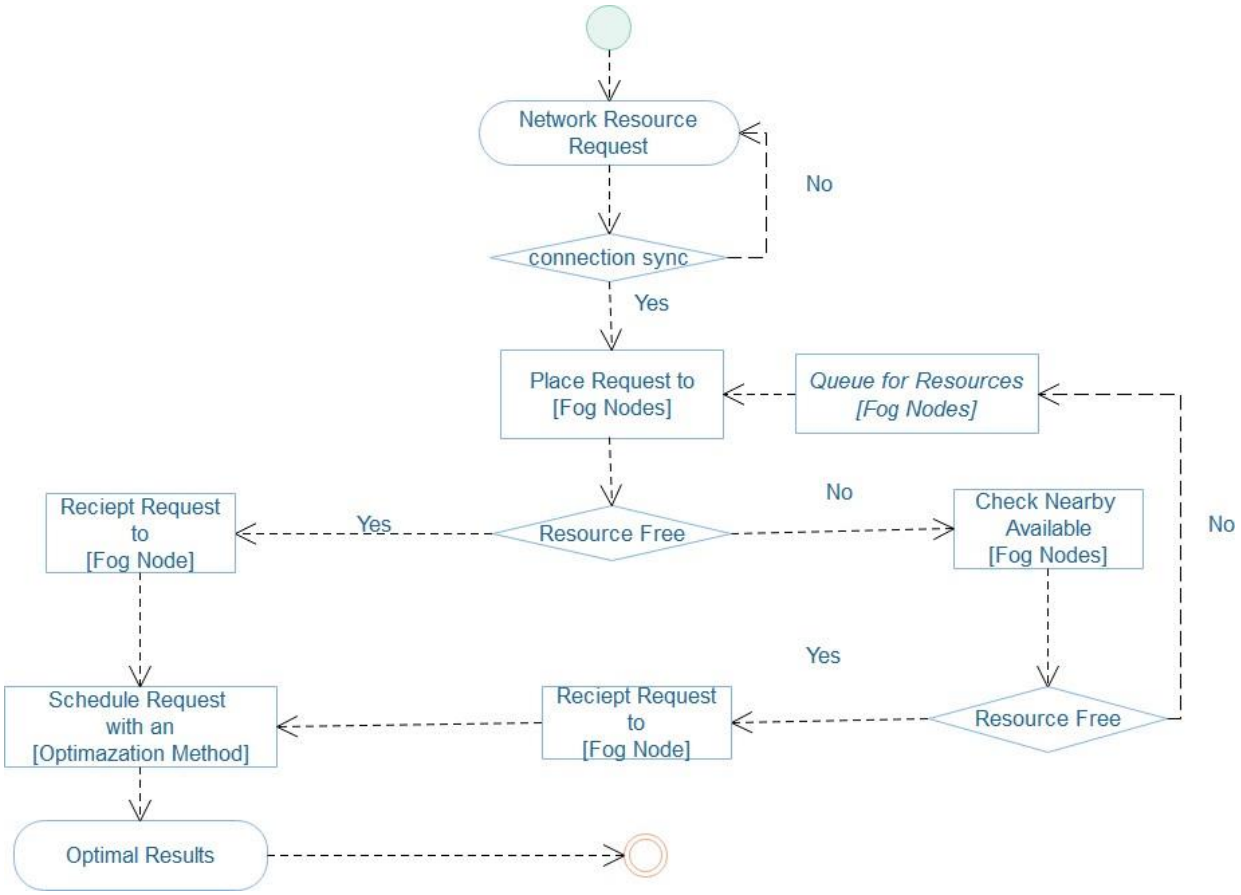

Figure 4.2: Workflow of the proposed framework.

In Fig. 4.2, an application initially sends a request to the nearby FN asking to use its services. First, the nodes synchronize connection with the nearby available FN. If the FN meets the demands of the nodes, the task is hosted, and the optimal scheduling method is applied to improve the overall performance of the fog network. However, if it does not meet the demand for some reason such as high demand for resource intensity, the node continues to search for an available FN. If there is no free FN, the tasks are queued until the resources are free. However, if the tasks are not delay sensitive, they are offloaded to the cloud servers until the terminal condition is met.

In the workflow shown in Fig. 4.2, the component FN receives the application requests as inputs and then generates the outputs that can be used to estimate the deployment factors such as energy consumption, network utilization, and service delay that can be experienced when processed at the edge on FNs. The set of constraints used to calculate the above three factors are mathematically modelled as shown in Eq. 4.1 to Eq. 4.3 below: In Eq. 4.1, m denotes a series of FNs offering services such as computing, network and storage:

$$T_{fn} = \sum_{i=1}^{m} (fn_i) \quad (4.1)$$

Where T_{fn} represents a total number of applications hosted. The FN hosts multiple applications that forms a series n which can be mathematically presented as shown in Eq. 4.2 below:

$$T_{App} = \sum_{j=1}^{n} (APP_j) \quad (4.2)$$

Where T_{App} represents a total number of application requests for a given system. Since each application request comprises a list of tasks that can be mathematically presented as shown in Eq. 4.3 below:

$$App_i = \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} \quad (4.3)$$

The resource constraints for each application request requirement is defined by the equation 4.4 to 4.6.

$$\sum_{i=1} A_{pp_i}^{RAM} \; t_{ij}a \leq R_f RAM \quad (4.4)$$

$$\sum_{i=1} A_{pp_i}^{MIPS} \; t_{ij}a \leq R_f MIPS \quad (4.5)$$

$$\sum_{i=1} A_{pp_i}^{Storage} \; t_{ij}a \leq R_f Storage \quad (4.5)$$

4.1 Assumptions

This section covers the information assumed before the mathematical modelling of the applications covered in this dissertation.

- The modelling is the first of its kind to be done in the network-planning problem, which means that there is no currently existing infrastructure.
- The connection between the sensor nodes and the FNs is always available, and the links' strength is reliable.
- Each FN has the same amount of vCPU, memory, and cost when installed in the fog network.
- The cloud is remotely located, far away from sensor nodes. Therefore, it only takes care of applications that cannot be served by the FNs. Moreover, the cloud is resource rich in terms of vCPU and memory.
- The FN periodically offloads its historical data to the cloud. This means after the FNs have done the required processing, in order to free the resources, data must be offloaded to resource-rich infrastructure, either for historical analyses or storage.

The focus and outcome of network-planning are based on the following:

- The behaviour of the FN resource consumption as tasks are pushed up to it.
- The optimal FN to which tasks can be pushed up, and the behaviour of network links.
- The capacity and number of FNs to be deployed for each use case.

4.2 Application Parameter Analysis

The experiments have been conducted by using the iFogSim simulation tool based in the scenario and model from Mahmud and Buyya (2018). The iFogSim has been selected because it is a well-established toolkit that runs on top of the CloudSim simulator, a widely used and tested cloud computing tool. Moreover, the iFogSim simulator permits the hierarchical composition of Fog devices, FNs and the cloud. It also supports analysis of application delays and has been widely used by Gupta et al. (2017).

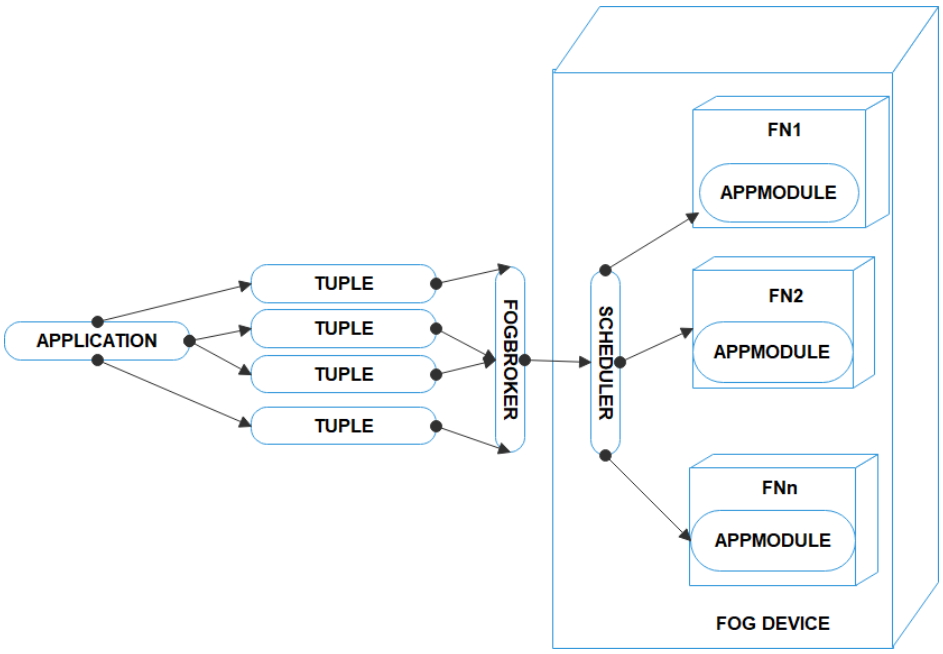

Figure 4.3: iFogSim Simulator Components.

Application: This allows researchers to design a Fog application based on the directed acyclic graph (DAG) model. The application is only complete with the following classes: *AppModule*, which represents the processing elements of Fog applications, and is analogous to a virtual machine (VM) in clouds. In this dissertation, the focus is mainly on this class. However, there are also *AppEdge* and *AppLoop* classes under the application. *FogDevice,* which is analogous to the datacentre in clouds. It complies with all the specifications for hardware of FNs and relational connections to others. The attributes accessible in this class are *memory*, *processor*, *network bandwidth*, and storage size. In this research, the scheduler method has also been implemented in this class. *Tuple*: This is a fundamental unit of communication between entities of the fog. It is analogous to cloudlets in clouds. Tuples are characterized by type and destination module.

4.3 Optimization and Modelling Approaches

Dealing with a network of heterogeneous devices serving applications is an NP-hard problem. It requires solutions to find the most efficient FNs on which to place incoming tasks. The best way to solve this problem is by using combinatory optimization methods, because FNs have limited resources. It is necessary to apply resource scheduling methods to avoid resource over-consumption on FNs which could lead to degradation of the quality of services (QoS). In order to achieve the best computation from the FNs, in this chapter, we present a comprehensive problem formulation using mathematical models that can be employed for a successful resource management framework. When sending tasks to FNs provided with a Fog network comprising applications, it is necessary to find the best application task placement in order to improve network performance. Furthermore, in order to have the best network services, mathematical modelling is adopted with the aim of addressing two main objectives, namely energy consumption and service delay, because offloading data to the edge should not reduce network reliability and make network design more complex. Rather it should be done in the interest of positive network design.

The process of transitioning from a given information scenario to mathematical modelling is known as mathematical problem formulation and it is used to define the entire range of constrains included in the scenario. The objectives are normally presented by formulae with the given constrains as variables. The objective of the modelling presented is to mitigate the energy consumption and service delay. To model the partitioning of the application into multiple tasks, a distributed data flow model is applied. For best FN performance two approaches have been applied, namely classical scheduling methods using linear-optimization approach and unsupervised machine learning clustering using non-linear-optimization approach. Moreover, we present the formulated model with different methods employed to find a solution for the multi–objective optimization problem.

The problem formulation presented in this dissertation focuses on the optimization of two objective metrics, namely energy consumption and service delay. Therefore, this is a multi-objective optimization problem that needs more than one solution to be evaluated in order to find an optimal solution. In such problems, more than one method can be a solution since one can either achieve minimized service delay at a high rate of energy consumption or vice versa. Therefore, in this section, we present five different methods that we hope will offer a solution to our problem.

4.4.1 Application Modelling with Classical Methods

The first classical methods extended to the distributed Fog computing topology is the First Come First Served (FCFS) Pardo et al. (2012), Mai et al. (2018).FCFS is the most popular task-scheduling method that requires tasks to be processed according to the order in which they arrive. The FCFS does not consider parameters such as task length or size which may be the varying factor that influences the network performance of the topology deployed. It takes the first task in the ready queue, regardless of any factor such as the length and size of the tasks. The second extended-scheduling algorithm is the Shortest Job First (SJF) which is a slightly modified version of the FCFS algorithm. SJF adds a feature considering the size and length of tasks. After sorting the tasks according to the parameters mentioned, they are then queued for the execution of the available FN in ascending order.

The third extended scheduling algorithm is known as Round Robin (RR) Devi and Uthariaraj (2016). This RR is designed to distribute time equally among tasks to access the FN resources. This time assigned to resources is known as the time quantum (usually 10-100 milliseconds), if the task is too long or big to be completed in a single access to FN resources, it will be stored back in the queue for the next access round. The final algorithm evaluated is Generalized-Priority (GP) Chekired et al. (2018), Zhang et al. (2018a). GP assigns fixed priorities to each task and the scheduler arranges the tasks in the ready queue, based on their priority (score) and executes the ones with the highest priority. The FNs offering services to these prioritized tasks are also prioritized according to their million-instructions-per-second (MIPS) value, which means that the FN with the highest MIPS size has high priority. The prioritization of tasks in this work is based on the execution time and task length.

After having applied the linear programming modelling approach to fog distributed networks, we have then used the unsupervised machine learning classification algorithm namely, K-Means Qi et al. (2018). The problem of the distributed fog topologies can be solved by using other alternative algorithms, which have been adopted in recent studies in the literature. However, the problem of identifying FNs that are resource efficient for different application task requests is challenging. Therefore, the best possible approach is grouping resource-efficient FNs according to clusters and then map tasks to clusters with adequate resources.

Algorithm 1 Selected Classical Internet of Things Application Algorithm

Require: *The Application Modelling of Four Classical Algorithms*

Initialization of Fog Broker and application
Add Application to the Fog Broker
for *i* ← 0 *to Number of Monitored Area* **do**
    for *j* ← 0 *to Number of Cameras Per Area* **do**
        Add Application to the Fog Broker
        FN ← *NodeName, MIPS, RAM, UpperBandwidth*
        LowerBandwidth, BusyPower, IdlePower
    end for
end for
Addmodules to the application model
Connecting the application modules in the application model with edges
Defining the input/output relationships of the application modules
Defining application loops to monitor the latency between loop modules
Initializing module mapping
Submit applications
Scheduler Algorithms in list (*FCSF, SJF, RR, PB*)
for *Each Virtual FogNode* **do**
    if *Module running in current Virtual FogNode* **then**
        return *Find next available Virtual FogNode*
    else
        Allocate module to Virtual FogNode
    end if
end for
Update energy consumption
Stop iFogSim simulation
Print the results

Therefore, the k-means algorithm is an unsupervised machine learning algorithm that is generally used for classification problems and is regarded as the most popular technique for clustering data.

4.4.2 Application Modelling with K-Means Clustering Method

This algorithm starts with random centroids and each object is assigned to the closest centroid as shown in Algorithm 2. The centroid values are then re-calculated based on the points of each cluster. This procedure continues until a termination criterion is met. Given that in the system design of a fog computing network, there are F sets of FNs containing m available resource coordinates to process set T of tuples (tasks) from the application and a set of K containing k clusters mathematically demonstrated by Eq. 4.4 to Eq. 4.6 as shown below:

$$\mathrm{F} = \{F_1, F_2, \dots, F_m\} \tag{4.4}$$

$$\mathrm{T} = \{T_1, T_2, \dots, T_m\} \tag{4.5}$$

$$\mathrm{K} = \{K_1, K_2, \dots, K_k\} \tag{4.6}$$

For each element in sets F, T and K there are (x, y) components representing the Tuple Execution Time (TET) and Tuple File Size (TFS) respectively. In order to measure the distances between tuples and the cluster's centroid, we use the Euclidean Distance (ED) formula as shown in Eq. 4.4 as:

$$\mathrm{ED} = \sqrt{(TET_i - TET_j)^2 + (TFS_i - TFS_j)^2} \tag{4.7}$$

The first objective when using K-Means is to define a number of set K suitable for Fog application. The idea is to get a K set that will yield the best possible network performance. In this work, the value of k is tested and chosen in terms of performance. After the first association of set F to set K, the new coordinates of resource clusters are periodically updated according to algorithm 2. Algorithm 2 below consists of two significant components, the FN and tuple. In the FN, the tuples are clustered based on parameters such as their file output size and application deadline. While in the fog device (datacentre) clustering, we have FNs clustered based on resource parameters such as memory and bandwidth. This is then used to calculate the network performance on the simulated applications. Finally, the tuples are mapped to FNs that can accommodate them with their available resources. The approach is as shown below in Algorithm 2.

Algorithm 2 Selected Machine Learning Application Clustering Algorithm K-Means

Require: *Performance Metric such as Energy Consumption*

Execution Time, Network Utilization
Generate Tuples in iFogsim
for *Each Tuple* ← *T* **do**
    Find the Deadline and Size of T
end for
Start KMeans clustering and divide them into K clusters
Randomly initialize centroid A and B
for *Each K in T* **do**
    Compute Euclidian Distance of K with A and B
    if *previous distance == new distance* **then**
        Stop iteration
    else
        Add T to minimum clustered distance
        Repeat compute centroids
        Initialize virtual FogNodes in ifogSim
    end if
end for
for *V irtual FogNode in F* **do**
    Find Memory and Bandwidth of F
    Start KMeans clustering and divide clusters into K clusters
    Randomly initialize centroids C and D
end for
for *K in F* **do**
    Compute Euclidean Distance of K with C and D
    if *previous distance == new distance* **then**
        Stop iteration
    else
        Add F to minimum clustered distance
        Repeat compute centroids
    end if
end for
create a key value pair using HashMap function
for *Each T in K* **do**
    assign best F to T
end for
update energy consumption
Stop iFogSim simulation
Print the results

4.4 Overview of Optimized Metrics

At least three QoS factors are considered in this work, including the network utilization, average energy consumption and service delay to answer question such as, given a distributed Fog network topology, which of the resource-scheduling methods can perform the best.

4.5.1 Network Utilization

In order to achieve efficiency in the fog network, a wise use of a specific design metric is significant, hence in this section, we discuss network utilization. In terms of modelling, network utilization, when extended to the distributed fog topologies, can be mathematically presented by Eq. 4.8 shown below:

$$Network\ Utilization = (TL \times TS)/(MST) \tag{4.8}$$

Where TL and TS denote the total delay and total size of the tasks, respectively. MST denotes the maximum simulation time.

4.5.2 Energy Consumption

In fog computing, energy is one of the network critical metrics that should be considered for efficient design. Therefore, this section presents the modelling of energy consumption. In this context, energy consumption refers to the overall energy consumed by the system and it is considered to be the most critical objective. This energy refers to the energy used by any component interacted from the sensor to the actuator loop. It can be mathematically presented by Eq. 4.9 as shown below:

$$Energy\ Consumption = CEC + (CT - LUUT) \times HLU \tag{4.9}$$

Where CEC represents current energy consumption, CT represents the current time, LUUT represents the last utilization update time and HLU represents the last host utilization.

4.5.3 Execution Time

In order to determine the system performance, it is sometimes necessary to know the time each task spends inside the CPU and that is known as the execution time. In this context, the execution time refers to the time the task spends on utilizing FN resources. It can be mathematically presented by Eq. 4.10 as shown below:

$$Execution\ Time = CT - SST \tag{4.10}$$

Where CT represents the current time and SST denotes the simulation start time.

4.5.4 Service Delay

Knowing how long the system application takes to complete tasks and send a response is one of the critical factors. This is known as service delay. In this context the service delay refers to the average processing delay of task sensing generated from sensors monitored at the edge placement for processing, and then back to actuating events. It can be mathematically presented by Eq. 4.11 as follows:

$$ServiceDelay = ST - ET \tag{4.11}$$

Where ST presents the simulation start time and ET denotes the simulation end time.

4.5 Chapter Summary

In this chapter, the proposed framework integration with classical and machine learning methods is presented. The goal is to have improved QoS on fog computing applications. This is done by scheduling various applications to FNs and then optimizing their performance, as well as network infrastructure by using the method presented. Furthermore, application dynamics in the testing environment are presented to show how the application components interact with the testing environment. Then we present the impact of the factors selected to determine the performance of each algorithm. This is followed by Chapter 5 where the classical methods modelled in this chapter are evaluated based on the performance metrics discussed. Then, classical and unsupervised clustering method are explored in this chapter to find the most optimal FN on which to place the application request is evaluated in Chapter 6.

Chapter 5

Evaluation and Analysis of Fog-Based Video Surveillance Application

5.1 Introduction

In this chapter, four distinct linear-optimization classical methods are evaluated. These evaluation processes are based on the components and discussions presented in Chapter 4. The tests comprise two stages, and in this chapter, we only evaluate stage one, which entails the classical methods. The effectiveness of the methods evaluated will help network planners to deploy applications in the fog landscape in order to manage and optimize network performance. Network performance is influenced by network metrics.

Therefore, the purpose of evaluating metrics is to provide measurable facts to determine if the outcome of the proposed framework is achieved. Questions focusing on the goals and objectives previously stated should be asked, for example, Are they successful? Are the objectives met through the proposed approach? In this chapter, we analyse the metrics stated previously which are network utilization presented by Eq. 4.8, energy consumption presented by Eq. 4.9, execution time presented by Eq. 4.10 and lastly, service delay presented by Eq. 4.11 in Chapter 4.

5.2 Simulation Tools and Parameters

The physical network topology configuration is assumed to have four different types of the device with the characteristics of a Raspberry Pi as shown in Table 5.1 Furthermore, the characteristics of communication links between the fog devices are shown in Table 5.3. The two communication links are 20 milliseconds and 100 milliseconds respectively. In Table 5.2 each smart camera comprises 1,600 MIPS (millions of instructions per second) and is connected to a Wi-Fi-gateway through a communication link of 10,000 Kbps bandwidth and 20-millisecond latency. The link between the gateway and the cloud has 10,000 Kbps bandwidth and 100 milliseconds latency.

Applied scheduling makes decisions before the application execution takes place. Therefore, the scheduling time of the actual CPU capacity is approximated because, at the time of scheduling, the scheduling methods must check if the free CPU capacity is adequate to handle each of the modules submitted by the application. For the application scenario presented in this dissertation, we have focused more on the module that needs to be executed and at most, the assigned CPU capacity values are as presented in Table 5.2 However, during the time of execution, each of the modules presented can use the CPU capacity differently, depending on the dynamicity of the module's receipt or transient relative to that of other modules that trigger the CPU use. These estimates come from the description of applications, which model the application as a workflow with direct graph attributes.

Table 5.1: Amount of CPU, RAM and Power for each device in simulation.

Device Type	CPU GHz	RAM(GB)	Power(W)
Cloud VM	3.0	4.0	107.339(M) 83.433(I)
Wi-Fi Gateway	3.0	4.0	107.339(M) 83.433(I)
Smart Camera	1.6	1.0	83.53(M) 82.433(I)
ISP Gateway	3.0	4.0	107.339(M) 83.433(I)

Table 5.2: Details of the size of each application module has

Tuple Type	CPU Length	N/W Length
Raw Video Stream	20 000	2000
Motion Video Stream	500	2000
Detected Object	1000	100
PTZ Params	100	100

Table 5.3: Communication capabilities of the designed network topology entities.

Source	Destination	Delay(ms)
Smart Camera	Wi-Fi Gateway	20
Wi-Fi Gateway	ISP Gateway	20
ISP Gateway	Cloud VM	100

5.3 Overview of Methodology and Network Design

The simulations were done for three states of mobile devices as in (Area, Mobile)= (1, 1), (1, 2), (1, 3) for different network topology configurations, namely: Config 1, Config 2, and Config 3 respectively. Each topology configuration was tested under the impact of four scheduling policies, namely, FCFS, SJF, GP, and RR. The impact of scheduling policies on the workflow of application task scheduling focuses on energy consumption, application latency, execution time, and network usage. To evaluate this project thoroughly, a simple evaluation setup is used to demonstrate how the modelling can be defined in terms of set boundaries, performance metrics and deployable devices constrains. The evaluation setup demonstrates a fog network topology accommodated by the proposed approach to computing framework. The network is designed to provide an analysis of the overall network performance. Fig. 5.1 presents an overview of the application-modelling problem setup in the simulation toolkit.

In the designed topology, the hierarchy consists of four layers, namely the sensors indicated as (s-0-1) and actuators (a-0-1), Fog cells indicated as (m-0-1) and (m-0-0), FN indicated as (d1) and the Cloud. In this work, we assume that the Cloud acts as an OpenStack environment hosted on our lab's private software-defined network (SDN). While the rest of the equipment's physical resource setups are assumed to be on Raspberry Pis 2B+. Moreover, on the topology demonstrated, the m-0-1 directly connects to d1, manages the task from s-0-1, and sends the responses to be acted upon by a-0-1. In Fig. 5.1, as the number of Fog cells (m-0-1) scales the number of FNs (d1) also scales, thus forming a Fog cluster orchestrated by the available FN.

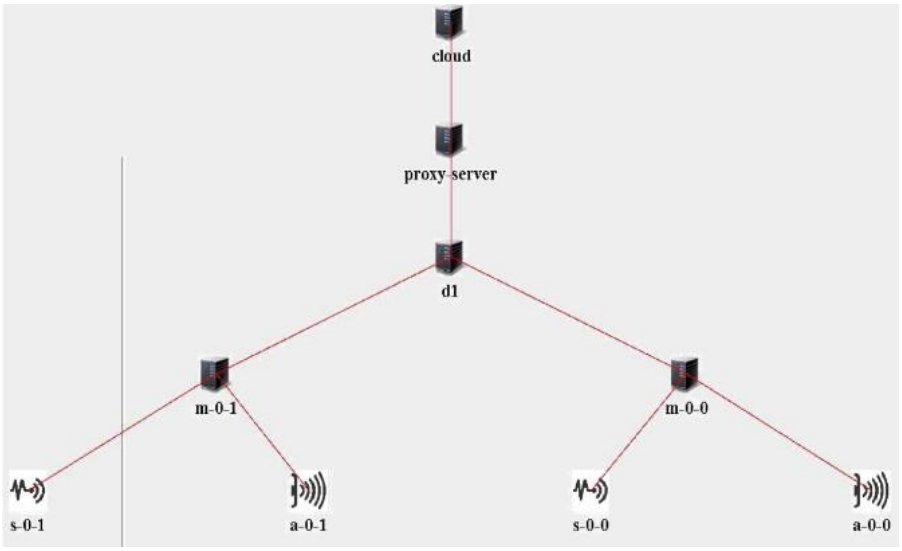

Figure 5.1: General Direct Acyclic Graph Setup Model.

Moreover, in Fig. 5.1 the FN (d1) processes the tasks from both Fog cells (m-0-1 and m-0-0) generated by connected sensors(s-0-1). The sensors act as modules representing features such as motion detection and fault detection. The network topology setup is a wireless local area network (LAN) connected to a powerful Lora-gateway connecting all the Raspberry Pis to the internet for further processing and computing. For framework evaluation purposes, each device in the designed topology has a unique name and identification number. The communication links vary depending on the type of connection to be achieved. In evaluation, we have chosen Video Surveillance. The choice of application is based on the capabilities that can demonstrate the implemented functionalities and mechanism of the framework clearly in terms of the defined objective metrics.

5.4 Experimental Application and Testing Environment

The testing was done on a Linux 18. 04 machine of the following specifications: Intel(R) Core (TM) i7 processor 3.20GHz x12, 40 GB RAM. The implementation was done on the previously discussed ifogsim simulator. The first experimental application was video surveillance shown in Fig. 5.2. In order to elaborate on how the resource management framework can be applied to real-time applications, we provide the dynamics of the modules in the video surveillance case study. The dynamics of the functional mechanism of each module application is shown Fig 5.2.

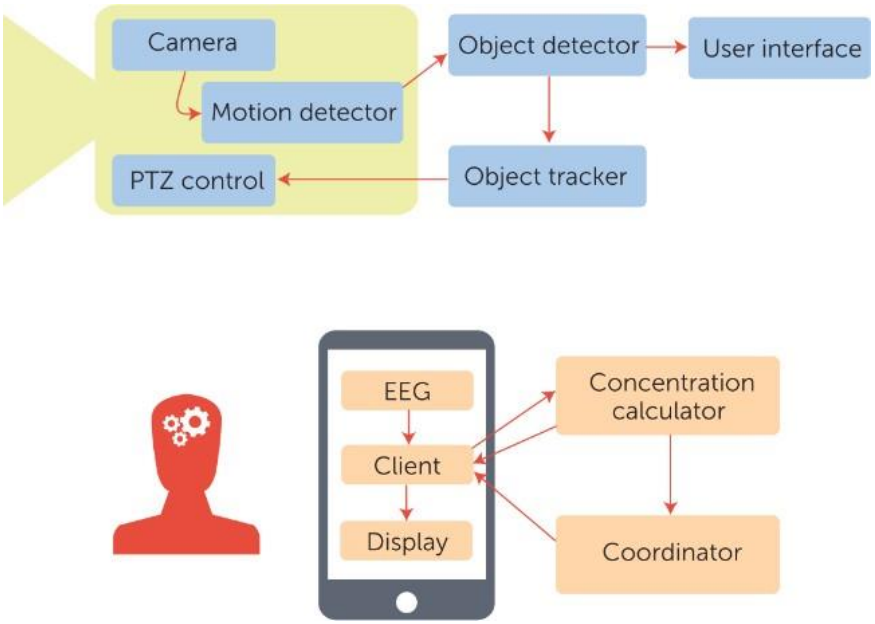

Figure 5.2: Video Surveillance Application Modules.

In Fig. 5.2 , the camera module takes video streams and feeds them to the motion detector module. The motion detector module filters the incoming video streams and pushes the filtered video streams to an object detector. The object detector module then identifies the object detected by calculating coordinates, and then pushes them to the object tracker. The object tracker computes the desirable PTZ of all cameras and optimizes them for clear capturing of the object. In this application design, it is assumed that the motion detector and PTZ control module are always placed in the cameras. The user interfaces on terminal devices, and continuously sends fractions of the application's live video streams containing each tracked object.

5.5 Results and Discussion

The goal of the implementation of the proposed framework is to find scheduling methods to install FNs that support all application processing at minimal consumption of energy used to compute, network and communicate the tasks across the network, and to achieve timely service delivery, which ensures the reliability of latency-sensitive applications. This problem is a multi-objective problem requiring multi-objective methods as a solution to achieve the competing objectives.

5.5.1 Average Energy Consumption

In the simulations undertaken in the first phase, the optimal solution was FCFS with a decrease of 0.0883 in the energy consumption percentage compared to that of SJF, which was the best after that of FCFS. Moreover, the worst energy consumption can be achieved by using the RR method at a percentage of 0.1388. The impact of the classical scheduling methods on the energy consumption and network performance is depicted in Fig. 5.3. Table 5.4 details the number of instructions in millions per second that each of the application modules has processed.

Table 5.4: Presents the effect of scheduling methods on network power.

Scheduling Method	**Average Energy Consumption (W)**	**Percentage (%) Difference**
First Come First Served	2840188.225	0.0
Shortest Job First	3076874.900	8.34 increase
Generalized Priority	3076872.200	8.33 increase
Round Robin	3220169.200	13.38 increase

The data in Table 5.4 is presented in Fig. 5.3. From the visualization in Fig. 5.3 shows that as the number of surveillance cameras monitoring the area increases, the number of Fog devices needed also increases, so the total energy consumed by the system also increases. Hence, Fig. 5.4, shows that it is clear that the FCFS is better than the SJF, RR and GP algorithms. However, among these other three less-optimal scheduling policies, the SJF, and GP scheduling policies performed almost the same as previously discussed in terms of percentage. This is presented in Table 5.4

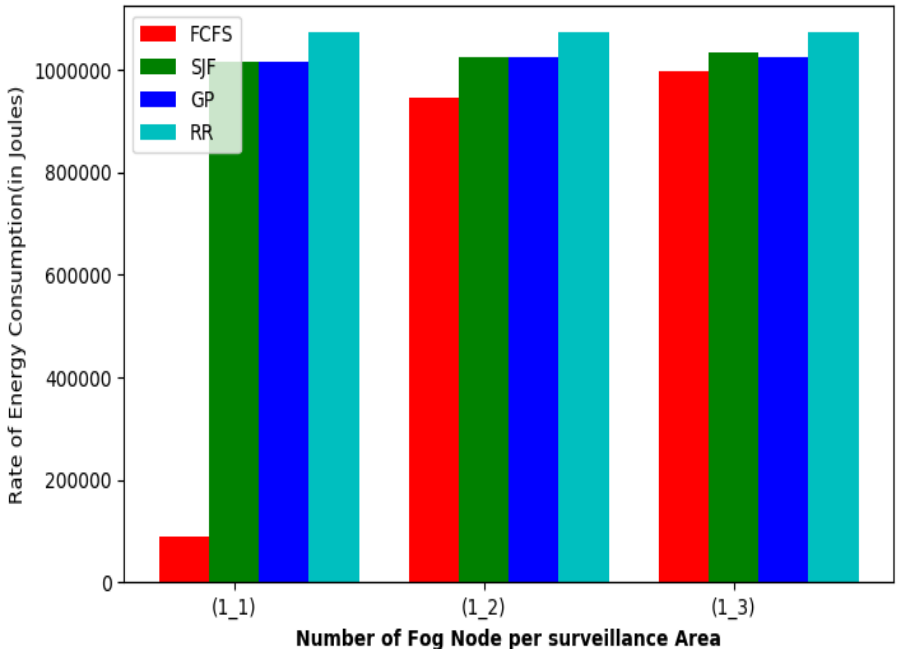

Figure 5.3: Average energy consumed by the system.

Moreover, to show clearly the percentage increases tabulated in Table 5.4 of the most optimal classical method, which is FCFS, relative to other evaluated methods, Fig. 5.4 below shows the results of the comparison.

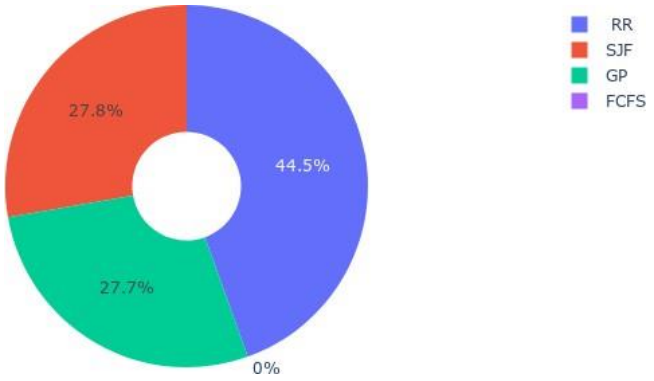

Figure 5.4: Percentage difference in the average energy consumed by the system.

5.5.2 Average Network Utilization

The second objective of the implemented framework is to minimize network utilization. According to Table 5.5, optimal performance of the framework can be achieved by implementing FCFS. The FCFS scheduling method shows a reduction of 0.74% in network utilization compared to SJF. It outperforms all methods in network utilization. Moreover, both GP and RR can be implemented as substitutes for FCFS after SJF. However, they both give worst-case performance of the framework. Table 5.5 details the number of instructions in millions per second that each application module processes.

Table 5.5: Presents the effect of scheduling methods on the network utilized by the system.

Scheduling Method	Network Utilization (KB)	Percentage (%) Difference
First Come First Serve	62766.00	0.0
Shortest Job First	62303.00	0.74 decrease
Generalized Priority	62470.80	0.47 decrease
Round Robin	62470.80	0.47 decrease

The percentage results in Table 5.5 show the relationship between the percentage increase from the most optimal classical method, which is FCFS, and the other evaluated methods. This can be explained further by Figure 5.5 below.

Furthermore Table 5.5 can be used to illustrate network utilization as shown in Fig. 5.6 which indicates that the increase in the number of cameras per area monitored is directly proportional to that of FNs, so is the increase in the number of network resources used. In Fig. 5.6, the RR-scheduling method appears to be optimal in the first network topology configuration. However, when the average network utilization is considered, the FCFS appears to be better than SJF, and GP with decreases of 0.74 and 0.47 per cent respectively. The illustration in Fig. 5.6 clearly shows that the FCFS is the best-case scenario on the framework for improved overall performance.

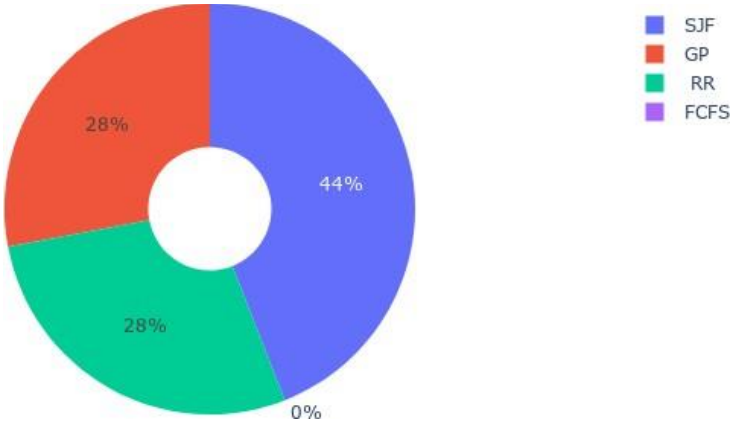

Figure 5.5: Percentage difference in the average network utilization by the system.

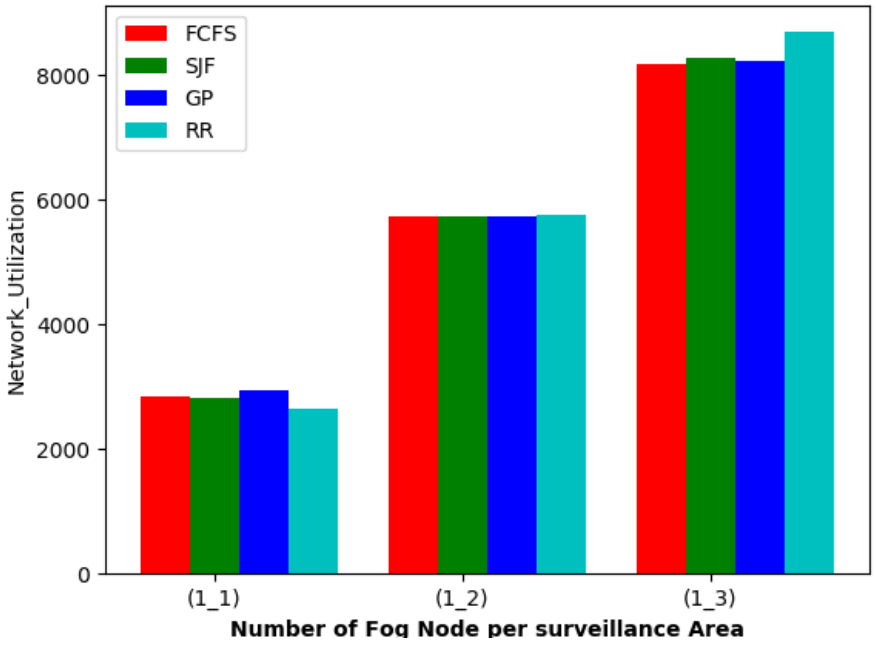

Figure 5.6: Average network utilized by the system.

5.5.3 Average Service Delay

The third objective concerns the implementation of the proposed framework to minimize the latency of application responses. According to Table 5.6 the framework's optimal performance can be achieved through the implementation of FCFS, and the second-best optimal scheduling method is RR at 0.1303 increase when evaluated against the FCFS scheduling method. Moreover, the worst performance of the framework is observed when both the GP and SJF methods are implemented as substitutes for RR and FCFS, following the proposed framework approach.

Table 5.6: Presents the effect of scheduling methods on the network service delay.

Scheduling Method	**Service Delay (ms)**	**Percentage (%) Difference**
First Come First Served	67.77	0.0
Shortest Job First	376.14	5.5 increase
Generalized Priority	376.14	5.5 increase
Round Robin	76.6	0.1303 increase

The results in percentages in Table 5.6 show the relationship of the percentage increase from the most optimal classical method, which is FCFS, relative to that of other evaluated methods. This can be explained further by Fig. 5.7 Below.

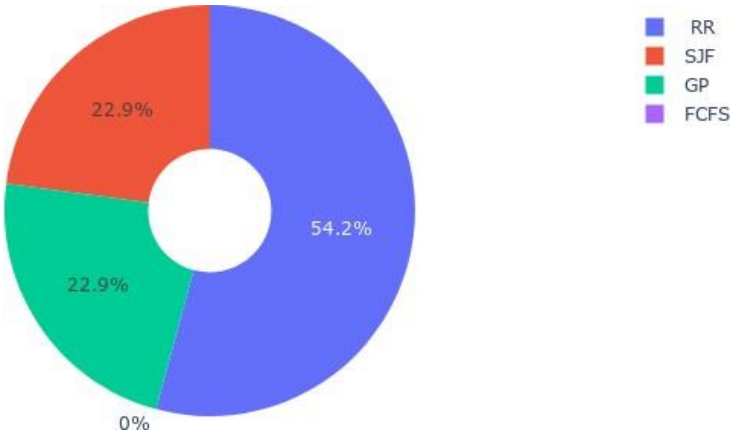

Figure 5.7: Percentage difference in average service delay for the system.

The illustration of the service delay shown in Fig. 5.8 shows that the increase in the number of cameras per area monitored is directly proportional to FNs, so is the increase in the number of network resources used. In Fig. 5.8, the RR scheduling method appears to be the second optimal method after the most optimal FCFS method. Moreover, Fig. 5.8 shows that the SJF and GP yield the worst performance. However, from the average service delay, it can be concluded that the number of surveillance cameras in both was increased by 5.5 percent. The illustration in Fig. 5.8 clearly shows that the best-case scenario implemented on the framework for improved overall performance is the FCFS.

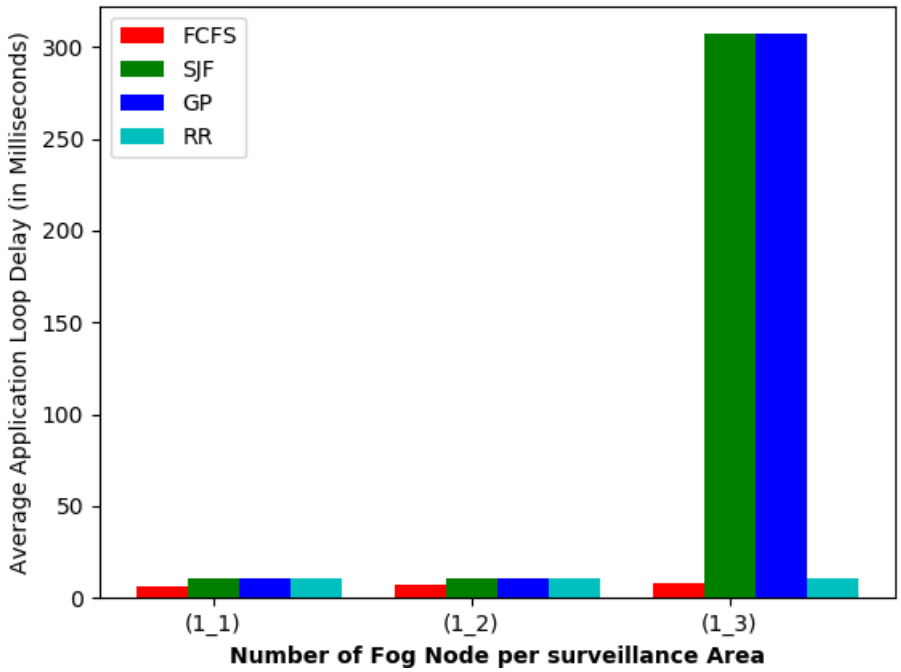

Figure 5.8: Average service delay experienced by the system.

5.5.4 Average Application Execution Time

The last objective metric investigated in this dissertation is the execution time. Table 5.7 below presents the impact of each method implemented in the framework. According to the results shown in Table 5.7, the FCFS method outperformed all the other scheduling methods. The possible technical reasoning for FCFS scheduling method to be optimal than other scheduling policies in terms of energy could be that, when FNs process tasks using FCFS it only accept what is given to it regardless of size or length. Thus, it does not consume more energy while sorting or arranging tasks. While on the other side the SJF, and GP needs to do the sorting and arrangement according to length of task and size of task respectively. In percentage terms, the SJF followed at 4.9% increase compared to that of the FCFS, which shows a huge gap between the methods evaluated. Moreover, we have the performance presented by the RR at a 5.5% increase. Finally, the worst-case scenario was presented by the GP at 5.8% from the FCFS, which is the worst optimal method of scheduling. Fig. 5.10 below illustrates the results from Table 5.7.

Table 5.7: Presents the effect of scheduling methods on the network execution time.

Scheduling Method	**Execution Time (ms)**	**Percentage (%) Difference**
First Come First Served	1283	0.0
Shortest Job First	7479	4.9 increase
Generalized Priority	8643	5.8 increase
Round Robin	7962	5.5 increase

The results in percentages in Table 5.7 show the relationship of the percentage increase from the most optimal classical method, which is FCFS, relative to that of other methods evaluated. This can be explained further by the Fig. 5.9.

Fig. 5.10 illustrates the impact of scheduling policies on the average execution time for various configurations that have been adopted to validate the proposed framework. From the experimental results, the FCFS method appears to be the most optimal compared to SJF, GP, and RR. However, in configuration 3, there was an increase in the number of surveillance cameras. The RR outperforms the SJF, and GP compared to configurations 1 and 2.

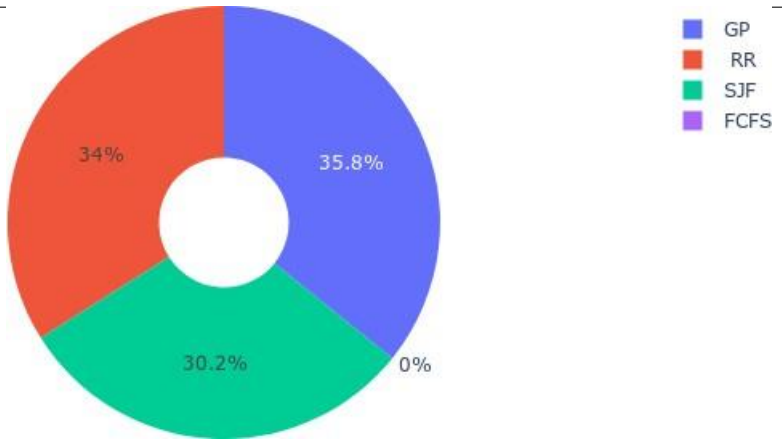

Figure 5.9: Percentage difference in the system CPU execution times.

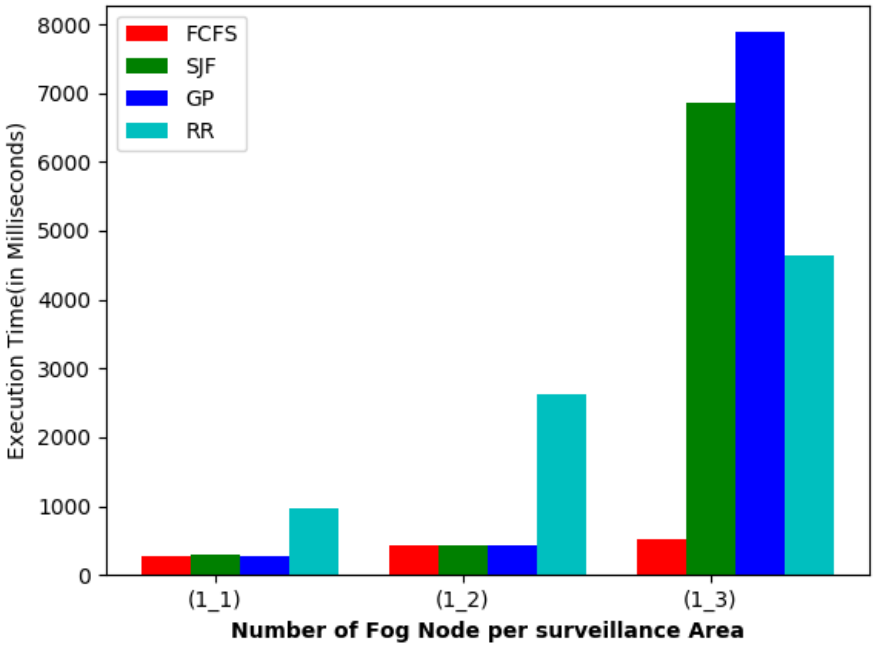

Figure 5.10: Average execution times of the system.

5.6 Chapter Summary

We have now evaluated the different LP scheduling methods used to validate the proposed framework and observed the impact each method has on the objective metrics. This dissertation focuses on small-scale Fog network application problems, but the same methods are also applicable to large-scale networks. The approach to the multi-objective problem has first been to observe the impact of each method implemented on a single objective metric. The functionality of each method and the result analyses are tabulated in Table 5.4, Table 5.5, Table 5.6, and Table 5.7. The approach of optimizing one objective metric aims to study the trade-off between the competing objectives such as service delay and energy consumed when each method is applied.

The results in the tables show the percentage differences between the single and multi-objective problems. The results of the competing objectives of energy consumption and latency for the FCFS are optimal. The GP optimal latency is increased by5.5% and the energy consumption is increased by8.33% by the optimal FCFS method for both competing objectives respectively. For the SJF method, the optimal latency is increased by 5.5% and the energy consumption is increased by 8.34 % compared to FCFS for both competing objectives respectively. Finally, the worst-case method is the RR with an increase of 13.03% in optimal latency and 13.38% in energy consumption compared to the FCFS method for both competing objectives respectively. The trade-off achieved for competing objectives such as the optimal latency and energy consumption for the GP method is 2.83%. For the SJF the trade-off is 2.84% while for the worst-case method, RR it is 0.35%.

Therefore, in summary the FCFS method is optimal overall and optimizes the performance of both competing objectives namely, optimal delay and energy consumption. The possible technical reasoning for FCFS scheduling method to be optimal than other scheduling methods in terms of energy consumption could be that, the FCFS scheduling method enables FNs to only process what is given to it regardless of size or length. Thus, it does not consume more energy while sorting or arranging tasks. While on the other side the SJF, and GP needs to do the sorting and arrangement according to length of task and size of task respectively. Since FCFS is an optimal classical method for competing objectives, the next step is Chapter 6 which presents the second stage of resource scheduling by using the clustering approach. Then the performance is evaluated against the FCFS.

Chapter 6

Evaluation and Analysis of Fog-based Smart Farming Application

6.1 Introduction

In this chapter, an unsupervised machine learning clustering method is implemented to improve the overall network performance of factors such as energy consumption, network utilization and service delay. This method is known as the K-means clustering algorithm and has been discussed in Chapter 4. The K-means algorithm has been used in studies Gu (2000), Qi et al. (2018) and has been identified as the best-performing clustering algorithm for data analysis and clustering. Hence, it has become the best method to provide optimal resource placement according to Garg and Trivedi (2014), Singh and Singh (2017) and Hatamlou et al. (2012).

In this chapter, the clustering method is applied to a *smart farming* use case. Based on the current state-of-the-art review applying K-means-based clustering to fog computing, smart farming has not been explored. For evaluation purposes, the proposed K-means algorithm is compared with the best-performing classical algorithm from Chapter 4. More precisely, we are searching for a solution that is generic and can be applied to optimize the performance of any fog network topology.

6.2 Simulation Tools and Parameters

The configuration of the physical network topology is the same with section 5.2 and the table 5.1, 5.2, and 5.3 are used for simulation.

6.3 Overview of Methodology and Network Design

The simulations were done for three farms and FNs as in (Farm, FN) = (1, 1), (1, 2), (1, 3) for different network topology configurations. Each topology configuration was tested under the impact of two scheduling methods, namely, FCFS which was the best performing one amongst the classical algorithms evaluated in Chapter 5, and the K-Means clustering proposed in this chapter. The evaluation setup demonstrates a fog network topology accommodated by the proposed computing framework approach. The network is designed to analyse the overall network performance. Fig. 6.1 presents an overview of the application modelling problem setup in the simulation toolkit.

In the designed topology, the hierarchy consists of four layers, namely, the sensors indicated as (s-0-1) and actuators (a-0-1), Fog cells indicated as (m-0-1) and (m-0-0), FN indicated as (d1) and the Cloud. In this work, we have assumed that the Cloud acts as an OpenStack environment hosted on our private software-defined network (SDN) lab. While the rest of the equipment's physical resource setups are assumed to be Raspberry Pis 2B+. Moreover, in the demonstrated topology, the m-0-1 connects direct to d1, manages the task from s-0-1, and sends the responses to be acted upon by a-0-1. In Figure 6.1, as the number of Fog cells (m-0-1) scales, the number of FNs (d1) also scales, thus forming a Fog cluster orchestrated by the available FN. The structure of the DAG setup for the smart farming simulation follows the setup in figure 5.1.

6.4 Experimental Application and Testing Environment

In this section, we present a smart farming application on Linux 18. 04, of the following specifications: Intel(R) Core (TM) i7 processor 3.20GHz x12, 40 GB RAM. It was implemented on the ifogsim simulator previously discussed and was modelled as shown in Fig. 6.2. The dynamics of each module application functional mechanism is provided below.

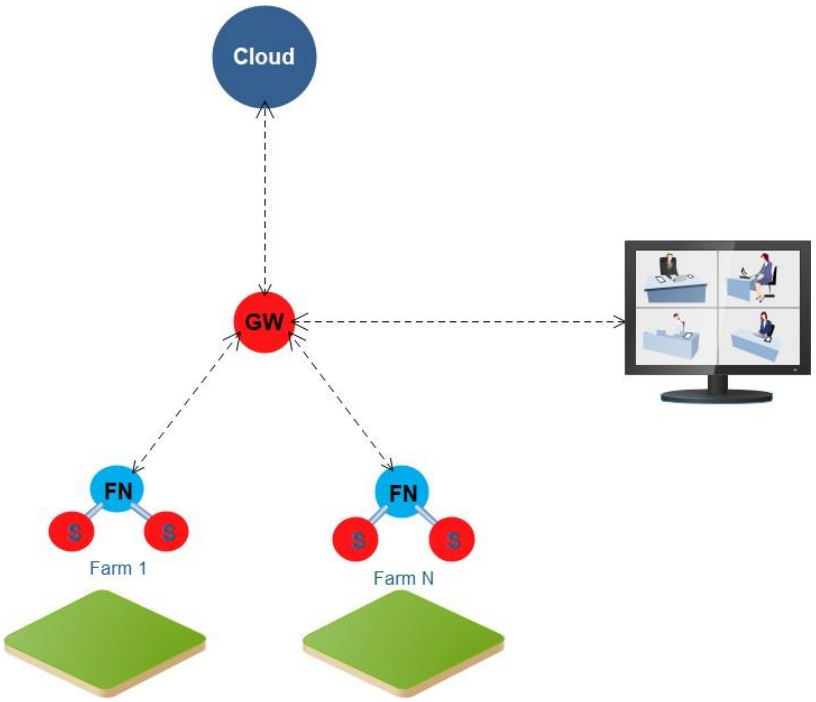

Figure 6.2: Design of Smart Farming.

The smart farming application in Fig. 6.2 above is assumed to be an autonomous system comprising four main components, namely smart sensors (S), FN(FN), gateway (GW), liquid crystal display (LCD) screen. The physically deployed smart sensors are responsible for watering plants; managing temperature, humidity, soil nutrients, water level, and pH level. All the sensors are connected to FN, which in turn connects to GW that communicates with the LCD screen for real-time analytics. The model approach is assumed as initially having each sensor device with a unique id communicate with the FN for task processing. The FN has algorithms for handling tasks on available resources. The analytics of each processing done are shown on the LCD screen.

6.5 Results and Discussion

In this chapter, the most optimal method from classical scheduling methods discussed in Chapter 5 is evaluated against the unsupervised machine learning clustering method K-Means. The main objective metrics are service delay and energy consumption. Therefore, the focus will be on the competition between these two objective metrics. The manipulation is based on the smart farming application in the fog computing environment. Finally, we present the summary of the best parts of each method and the types of application recommended for each method.

6.5.1 Energy Consumption (EC) vs Service Delay (SD)

The data in Table 6.4 represents the performance of each objective metric under the two methods evaluated, namely, FCFS and K-Means. However, network utilization and execution time objectives are later evaluated as well. In the data represented by Table 6.4, the first column indicates the (farm, FN) correspondence of the experimented use case instances. While the second and third columns present the recorded result of each method implemented to evaluate the performance of the objective metric. In the evaluation process, each implemented method is analysed on the impact it has on the objective metric. However, the results table are tabulated with respect to importance of the competing metric.

Table 6.4: Presents the effect of scheduling methods on the network power.

Cases	**FCFS**		**K-Means**	
Problem	**Service Delay (ms)**	**Energy Consumed (W)**	**Service Delay (ms)**	**Energy Consumed(W)**
(1,1)	21.21	895656.585	116	896010.389
(1,2)	22.92	946729.41	114	946644.857
(1,3)	30.40	997802.23	122	992747.88
Average	**24.84**	**946729.41**	**117.30**	**945134.38**

The results tabulated in Table 6.4 are depicted in Fig. 6.3 which shows that as the number of FNs deployed increases, so the energy consumed increases too. However, the K-Means method can maintain the optimal energy consumed. Therefore, it consistently outperforms the FCFS method. In percentage terms, the K-means is at 5.48%, while FCFS is at 5.70%. This implies that K-Means is 0.22% better than FCFS in terms of energy consumption. In our experiments, smaller percentage denotes better performance.

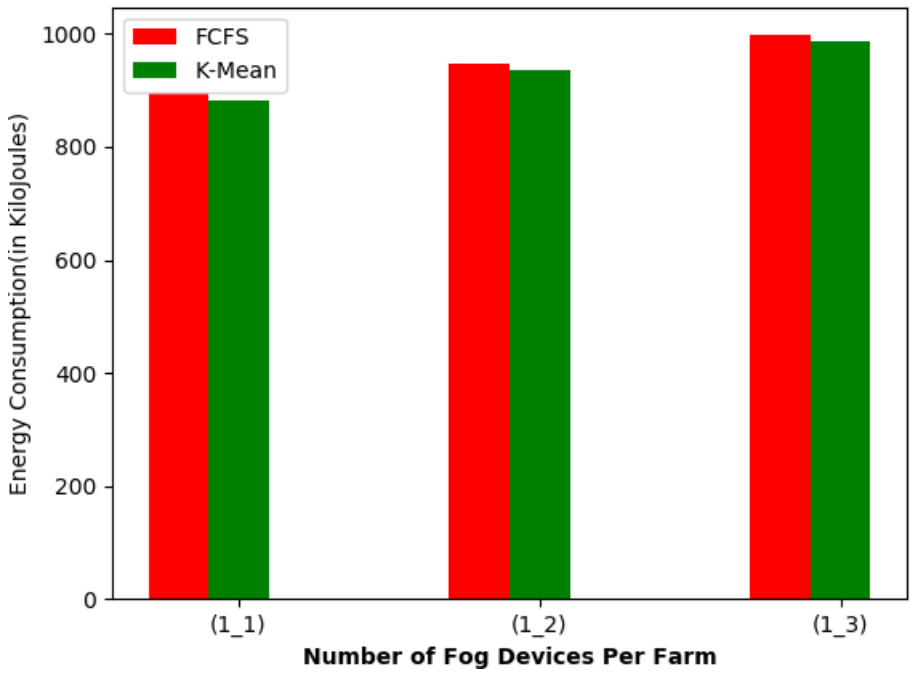

Figure 6.3: Average energy consumed by the system.

However, in terms of latency, the FCFS is at 1.72 %, while the K-Means is at 8.06%, meaning FCFS in terms of latency is 6.34% better than K-Means. Fig. 6.4 depicts the evaluation of service delay with the implementation of the two methods.

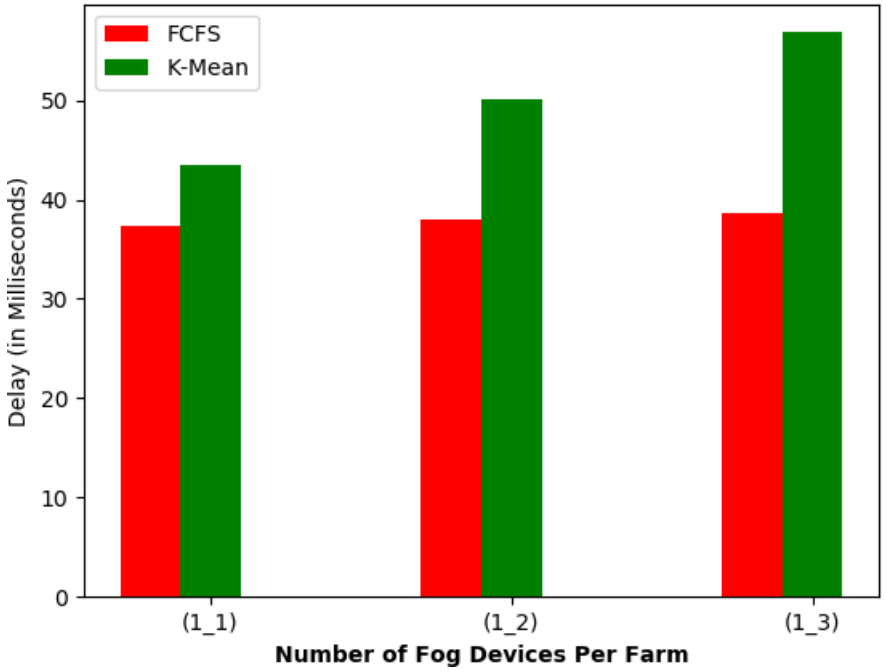

Figure 6.4: Average service delay experienced by the system.

6.5.2 Network Utilization (NU) vs Execution Time (ET)

The other two metrics evaluated and recorded in this section are the execution time and network utilization as shown in Table 6.5. These results show that the execution time of the FCFS method is 427.67 on average, while the network utilization of K-Means is 437492 on average. The results in table are depicted in Fig. 6.5 and Fig. 6.5

Table 6.5: Presents the effect of scheduling methods on the network utilities.

Cases	**FCFS**		**K-Means**	
Problem	**NU (Kb)**	**ET (ms)**	**NU(Kb)**	**ET (ms)**
(1,1)	10731.80	305	10375.80	96539
(1,2)	20927.60	448	20815.60	355067
(1,3)	31107.40	530	31055.40	860870
Average	**20922.27**	**427.67**	**20748.93**	**437492**

Fig. 6.5 shows that as the number of Fog devices per area increases so does the total network utilized by the Fog system. FCFS needs a decrease of 9473.37, which is 45.28%, to be equal to K-means.

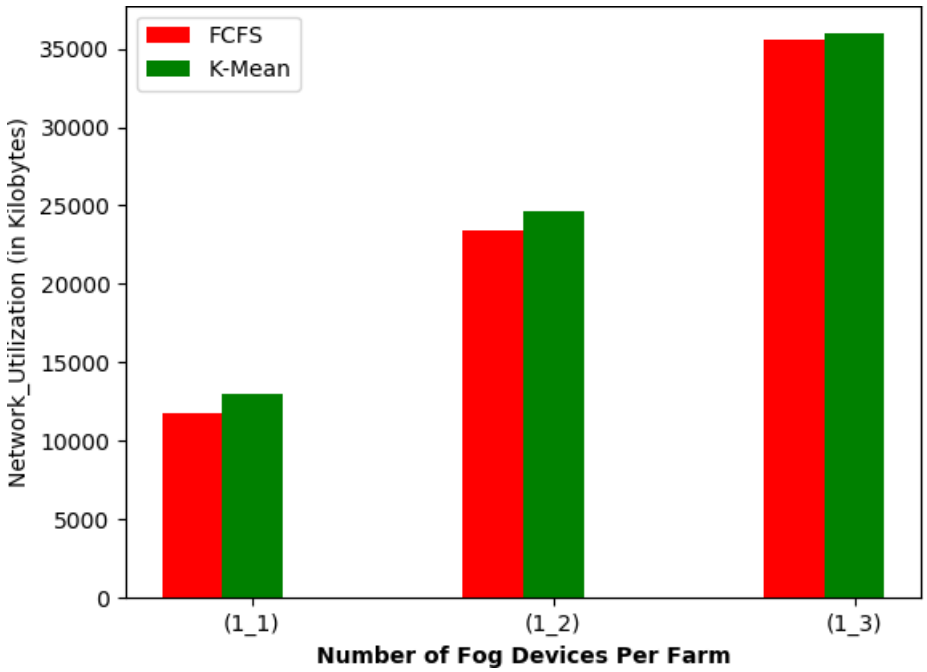

Figure 6.5: Average network utilized in the system.

However, as regards the total time the application modules spent on CPU for execution, the K-means needs a drastic decrease of at least 1022 times as shown in Fig. 6.6.

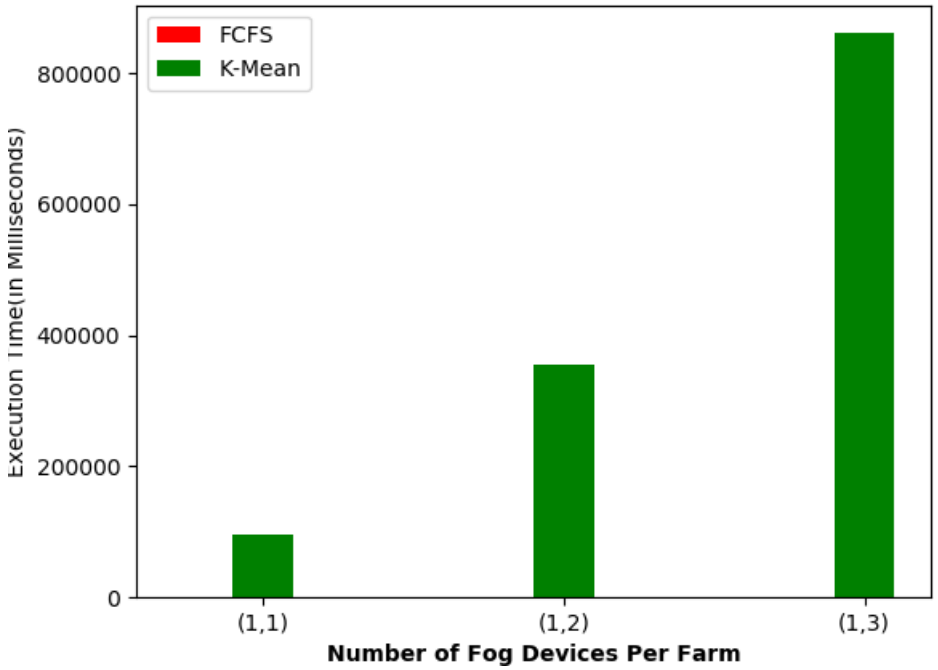

Figure 6.6: Average service delay experienced by the system.

To understand the competition between the main competing objectives' latency and energy consumption, which is also the focus of this dissertation, we have presented a trade-off analysis study tabulated in Table 6.6

Table 6.6: The effect of scheduling methods on the network power.

Cases	**FCFS**		**K-Means**		**Trade-Off**	
Problem	**SD (%)**	**EC (%)**	**SD (%)**	**EC (%)**	**FCFS**	**K-Means**
(1,1)	0.0	0.0	0.0	0.0	0.0	0.0
(1,2)	8.06	5.7	1.72	5.65	2.36	3.93
(1,3)	11.46	11.4	5.17	10.8	0.06	5.63
Average	**6.50**	**5.70**	**2.29**	**5.48**	**0.81**	**3.19**

In Table 6.6 above, two methods are evaluated based on three application scenarios for the competing objectives metrics of service delay, denoted as SD, and energy consumption, denoted as EC. In the table 6.6, the method with the smallest average percentage is the optimal one. The average percentage for service delay under the FCFS method is 6.50 % while the K-Means, it is 2.29% meaning the FCFS is 4.21% optimal in terms of service delay. The average percentage in terms of energy consumed is 5.70% for the FCFS method while it is 5.48% for K-Means. This means that the FCFS needs a 0.22 % increase in order to outperform the efficiency of the K-Means method in terms of the energy consumed by the system. Moreover, as regards the average trade-off between the competing objectives, the FCFS is at 0.81% while the K-Means is at 3.19%. This means FCFS is 3.01% more efficient than K-Means in spreading between the competing objectives. Fig. 6.7 and Fig. 6.8 below present the trade-off analysis of the competing objectives.

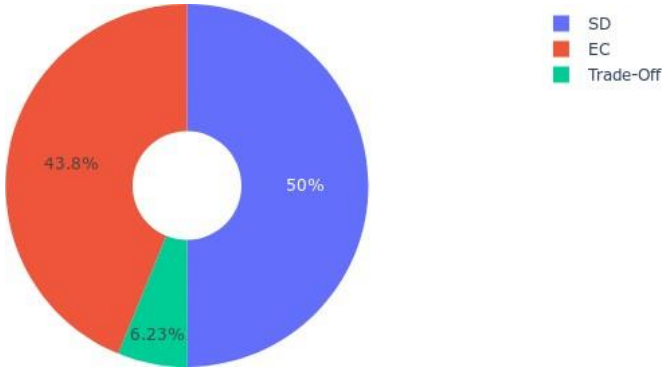

Figure 6.7: Trade-off analysis between objective metrics under FCFS.

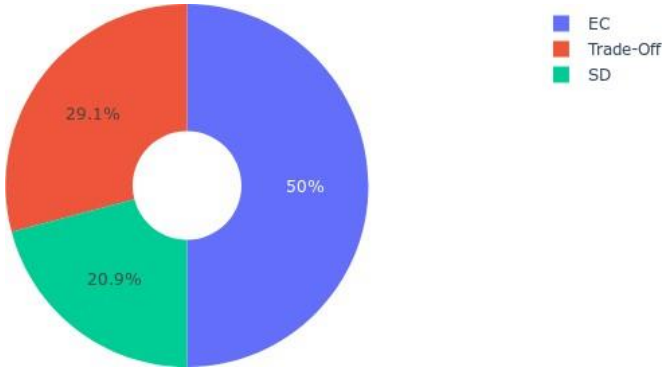

Figure 6.8: Trade-off analysis between objective metrics under K-Mean.

6.5.3 Comparison of the Task CPU Execution Delay

We have now compared all the methods selected and then the optimal ones only, by measuring their performance based on objective metrics such as energy consumption and service delay for different application use cases but we have not considered the inter-transmission time of application tasks following a deterministic distribution as designed by the simulation tool. Various tasks make up the entire application, and in this section, we measure the time spent by each task module of the application on utilizing the CPU resources. Various methods can be adopted to measure the inter-transmission and processing of the application modules. One way chosen in this dissertation is to measure the time of each module and depict it. In the video surveillance application simulated, there are at least four independent modules that complete the application as a system. This module comprises motion video stream, detect object, object location, and camera. These are presented as MVS, DO, OL, and C respectively on Table 6.7.

Table 6.7 presents the effect of scheduling methods on the time each task has spent on using the CPU resource.

(1,1)					
Method	**MVS**	**DO**	**OL**	**C**	**Avg.**
FCFS	0.81	0.11	0.46	2.10	**0.87**
SJF	5.0	0.60	0.81	2.10	**2.13**
GP	5.0	0.60	0.81	2.10	**2.13**
RR	5.0	144.54	0.81	5.0	**38.84**
K-Mean	61.89	49.31	30.30	5.78	**36.82**
(1,2)					
FCFS	1.53	0.11	0.81	2.10	**1.14**
SJF	10.0	0.60	1.53	2.10	**3.56**
GP	10.0	0.60	1.53	2.10	**3.56**
RR	5.0	131.61	1.53	5.0	**35.79**
K-Mean	57.60	4.31	33.70	6.04	**25.41**
(1,3)					
FCFS	2.24	0.11	1.17	2.10	**1.41**
SJF	502.64	0.60	2.24	2.10	**126.90**
GP	502.64	0.61	2.24	2.10	**126.91**
RR	5.0	34.35	2.24	5.0	**11.65**
K-Mean	118.36	30.32	24.92	5.78	**44.85**

In the previous sections, we have presented the overall execution time that an application takes to utilize the resources. However, we have not depicted the time each application module spent on utilizing the resources. Therefore, in this section, we show the time spent by the application modules in CPU when the presented methods as used as recorded in Table 6.7, which presents the average inter-processing time for the whole application. Fig. 6.9 shows the performance of each method when the motion video stream module is scheduled.

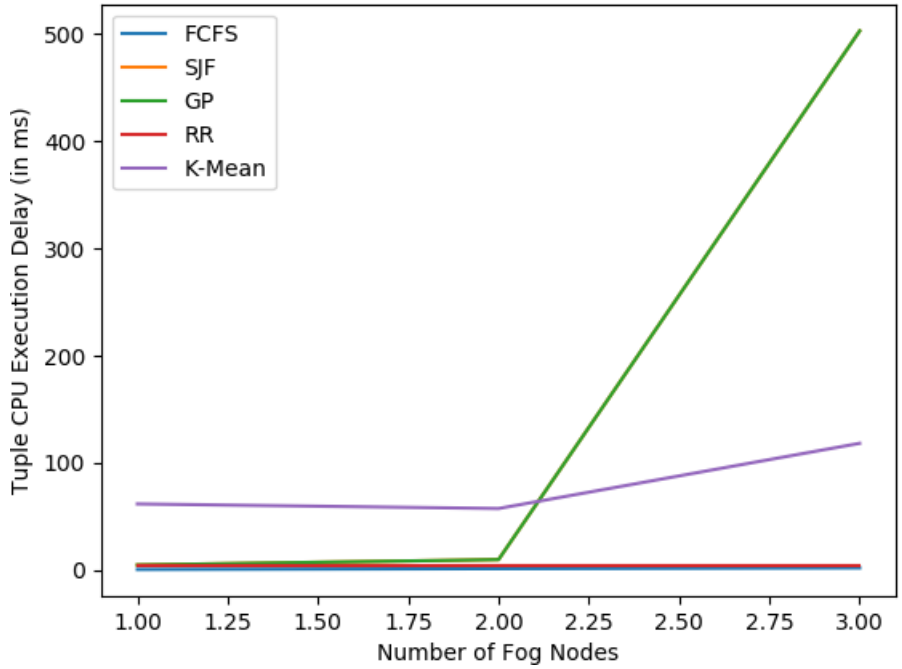

Figure 6.9: Video motion streams performance module comparison using scheduling methods.

Fig. 6.9 shows that the FCFS have spent the minimal time inside CPU for all the instances. This is followed by SJF and GP which have spent the same amount of time on using the CPU resources. However, just after 2, there is a sharp increase in the amount of time for GP which may be caused by an application with greater length and less priority, while the SJF remains consistent. Moreover, we have the RR method showing consistent accessing of the CPU resources which makes sense since the method allocates every task an equal amount of time to access CPU resources. Before 2, the application modules scheduled using the K-means method to use the CPU resource appeared to be the worst case scenario, which may be caused by the fact that K- means has spent more time clustering the modules according to length and time. Finally, after 2, K-means has performed better than the GP method.

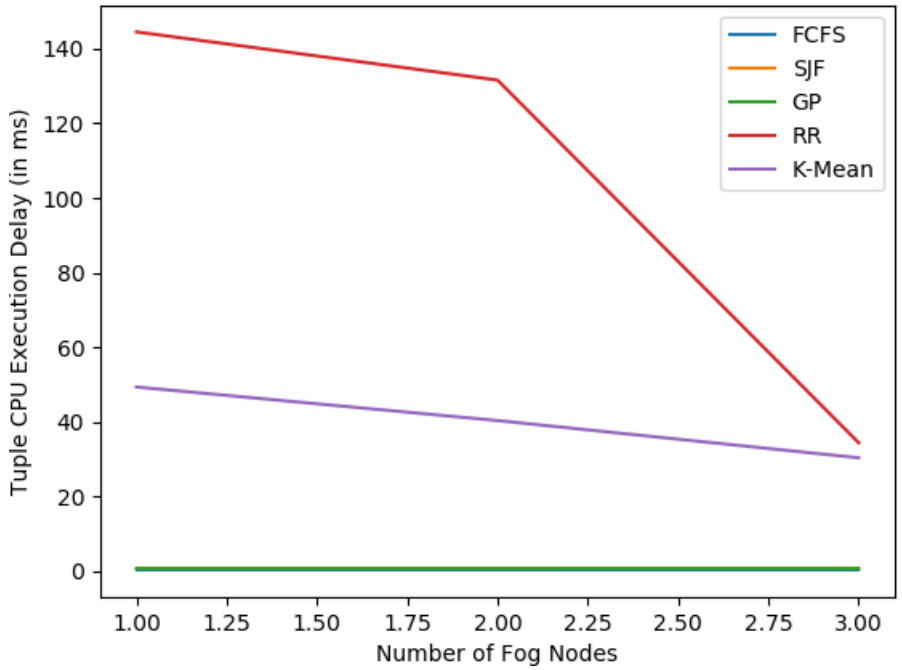

Figure 6.10: Detect object module performance comparison using scheduling methods.

Fig. 6.10 shows interesting results where the FCFS, SJF, and GP are shown to have spent the same amount of time, which happens to be the minimal time spent inside CPU, for all the instances. This may be only possible when each module has the same length. Regardless of priority, and moment of arrival on the CPU, it is still going to use the same amount of time for execution. However, the RR method has performed worse than the K-means method, which is confusing, in view of the consistency of the other three classical methods.

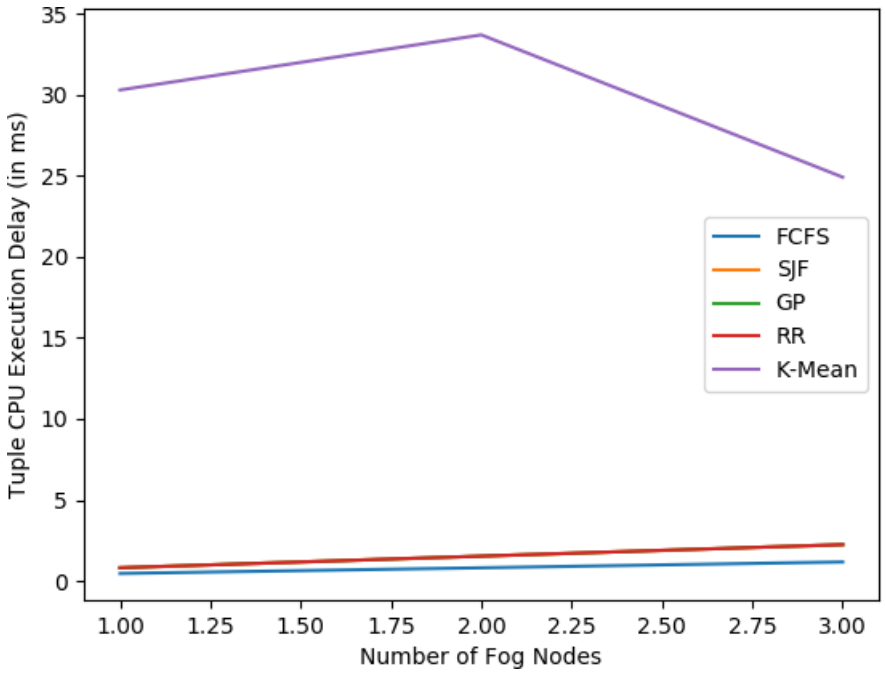

Figure 6.11: Object location performance module comparison using scheduling methods. Fig. 6.11 shows the K-means starting with a worst performance for small- scale FNs. However, as the FNs increase, they perform better. This shows that for larger-scale FNs the K-means is the best scheduling method. On the other hand, the FCFS, SJF and GP are shown to have spent the same amount of time, which happens to be the minimal time spent, inside the CPU for all the instances. It is possible that when all the modules are of the same length, regardless of priority and moment of arrival on the CPU, they will still use the same amount of time for execution. However, the behaviour of the RR method is shown to be proportional to the number of FNs.

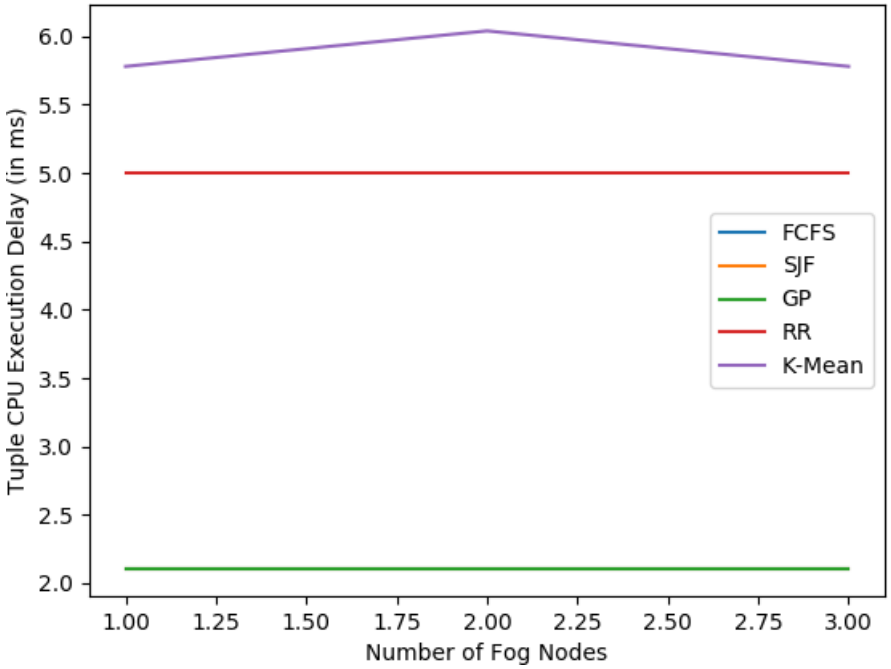

Figure 6.12: Sensor module performance comparison using scheduling methods.

6.6 Chapter Summary

In summary, on average the results show that the K-Means is the best method when compared to FCFS for applications such as smart farming. According to B. Chen et al .(2018), K-means reduce complexity by grouping data of same set together removing all the dissimilarities, and is also able to adapt with the fast-changing data traffic thus can suitably solve growth complexity than classical scheduling methods such as FCFS. Although it shows more service delay, the optimized energy usage means that the smart farming application will have long-lasting network reliability, which limits the risk of network breakdown as the FNs are guaranteed to be energy efficient in most cases. Furthermore, for a large farm with more Fog devices deployed, the K-means will clearly provide efficient network utilization, thus improving the overall performance of the smart farming application. As regards service delay, there are still optimization methods that can either be combined or used as stand-alone for better end-to-end delay performance. The overall result shows that K-means has the potential to avoid resource over-consumption in fog computing applications where network reliability is more critical than metrics such as service delay.

Chapter 7

Results Discussions

The goal of this study was to design and develop a resource management framework that could help network operators or providers to implement an optimal fog computing network, especially for mission-critical applications.

Designing such a framework comes with complexities. Various significant research studies have attempted to find methods to design a framework with optimal overall network performance. To summarise, for evaluation purposes, most framework development research makes use of optimizing methods to test the reliability and performance of the fog network. A significant finding from the results of this dissertation research shows that making use of multi-objective optimization methods to optimize the network performance of fog applications such as video surveillance is one of the best solutions. This concurs with the discussions of results by Santos et al. (2017), where a linear-optimization method is implemented to a foggy framework to evaluate the network performance when applied to a video surveillance application. This also explains why Bittencourt et al. (2017) have made use of classical methods to improve the performance of QoS in the task placement to FNs.

In this dissertation, the linear-optimization approach using classical methods perform efficiently for small-scale topologies, especially to mitigate the over-consumption of resources such as energy, and optimizing the service delay performance. With reference to research by Natesha and Guddeti (2018) where they have used a First-Fit linear-optimization method to find the resource efficient FN to host tasks from different applications as an approach to minimize latency, network utilization and energy-consumption , in this dissertation, we explain clearly why they have used such methods and our findings concerning performance are aligned with theirs. This dissertation's findings are almost similar to those of a recent study by Mai et al. (2018) which shows that linear-optimization methods can optimize the network performance, in particular service delays.

The findings have also shown that as the number of problems grows, so does the network complexity, which results in metrics, such as service delay, increasing non-linearly. The previous findings concurs with the findings from research work by karbizadeh et al. (2018) where they scaled the number of FNs to 22, the observation was that as the number of FNs increases, each device needs energy so is the energy-consumed by the entire network system. However, making use of algorithms such as Genetic Algorithms helps minimizing resource consumption. According to Zhu et al. (2019), another method one can use to deal with such complexities is clustering concept. This claim arises from the fact that the author

has used clustering methods to optimize the performance of the Folo framework. It is therefore wise to use this dissertation's findings as evidence, since it concurs with this claim, mainly because when the task clustering and scheduling method was applied in a smart farming application, the resource efficiency on FNs was better when compared to linear optimization methods. From the above discussions, it is clear that fog application deployment will soon be the potential solution due to its efficiency in resource usage and optimal responses. Therefore, the next chapter summarises the dissertation and describes improvements for future work to explore.

Chapter 8

Summary and Conclusions

Fog computing is designed to provide service closer to EU devices at the edge of the network. Therefore, it is crucial to design an efficient network to provide the best services. To do so, this dissertation presents a design for a reliable framework approach to designing and finding the most optimal performance of FCNs through the scheduling method that can be adopted for the proposed framework to optimize objective metrics such as energy consumption, network utilization, service delay and CPU execution time. The optimization approach focuses more on the two Fog network design objective metrics, including energy consumption and service delay. To investigate the approach that serves performance best, we have considered input parameters such as memory, vCPU, bandwidth, and network link speeds of both the FNs and the EU devices. The framework entails methods to ensure the optimization of the objective metrics without exhausting resource-constrained devices.

Before the development of the proposed framework, in the course of this dissertation, a comprehensive review of the background and related studies focusing on the scientific and technical analysis of approaches that can be applied to design and implementation of a framework in the fog computing landscape have been considered. The background involves the introduction of IoT, fog computing, technologies such as virtualization, the concepts of resource management and resource-scheduling and offloading methods. Moreover, problem formulation, fog computing use-cases and simulation environments and mathematical modelling have been explored and applied where necessary. Before, applying the proposed framework to solve the existing fog network problems, different problems have been designed with necessary input parameters and network constraints, including FNs, network links, and task modules. The initial task placements have first been randomized and pushed to the available FNs. The detailed problem description has been modelled and presented by following the proposed approach. The framework focus comprises two key objectives metrics, and five multi-objective methods have been evaluated to find optimal solutions. More precisely, the First Come First Served, Shortest Job First, Round-Robin, Generalized-Priority, and K-Means methods have been implemented to solve the resource management optimization problem through the proposed framework approach.

From the results obtained, the generic observation of the trends of objective

metrics, including energy consumption and service delay, is that as the complexity of the network increases so does the objective metric. However, for the same complexity when using the clustering technique, the energy consumption objective decreases while the service delay increases. For example, when the FCFS method is compared to the K-Means clustering method, the combined performance of the objective metric energy consumption is 0.22% optimal while the network utilization is 45.28% optimal, whereas, for the same complexity, the FCFS method dominates by 4.21% in terms of service delay and exponentially in terms of CPU execution time. In terms of trade-off analysis, the FCFS achieved the most optimal results at 3.01% better than the K-Means method. The results show that this is a multi-objective problem comprising competing objectives with different optimization methods. Therefore, the design of the optimal FCN topology for small-scale problems is subject to the requirements of the application. This simply means that the designers can make choices based on the objective metric, whether it is energy consumed mitigation or service latency. In this dissertation, the K-Means method has achieved the most optimal results in terms of network utilization and energy consumption and is also promising for large-scale networks.

8.1 Future Research Work

To the best of our knowledge, the resource management framework development in fog computing is still a researchable topic. Hence, the work in this dissertation can be extended to various research fields, including some of those mention below:

- In this dissertation, we have focused more on the overall application loop service delay. However, for better modelling and approximation of the entire service delay, the module queuing and FN data processing could make a major contribution if considered for improvement. However, the proposed framework does not consider cost optimization methods for designing cost-effective network application deployments.
- Another problem with the proposed framework is the fact that it has not considered the task module handover across the aggregated FNs. In addition, this dissertation focus is mainly on small-scale networks, hence the research focus can be extended to large complex network and make use of machine-learning concept for resource management. Finally, this dissertation only considers optimal placement of task rather than optimal placement of FNs. Hence this is another direction researchers can explore.

Bibliography

Dr Agarwal, Saloni Jain, and Others. Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv Prepr. arXiv1404.2076, 2014.

Kevin Asthon. That ' Internet of Things ' Thing. RFID J., page 4986, 2010. ISSN 00280836. doi: 10.1038/nature03475. URL http://www.rfidjournal.com/article/ print/4986.

Kay Bierzynski, Antonio Escobar, and Matthias Eberl. Cloud, fog and edge: Cooperation for the future? 2017 2nd Int. Conf. Fog Mob. Edge Comput. FMEC 2017, pages 62–67, 2017. doi: 10.1109/FMEC.2017.7946409.

Luiz F Bittencourt, Javier Diaz-Montes, Rajkumar Buyya, Omer F Rana, and Manish Parashar. Mobility-Aware Application Scheduling in Fog Computing. IEEE Cloud Comput., 4(2):26–35, 2017. ISSN 23256095. doi: 10.1109/MCC.2017.27.

Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. Proc. first Ed. MCC Work. Mob. cloud Comput. - MCC '12, (March):13, 2012. ISSN 978-1-4503-1519-7. doi: 10.1145/2342509.2342513. URL http://dl.acm.org/citation.cfm?doid=2342509.2342513.

Antonio Brogi, Stefano Forti, and Ahmad Ibrahim. How to Best Deploy Your Fog Applications, Probably. Proc. - 2017 IEEE 1st Int. Conf. Fog Edge Comput. ICFEC 2017, pages 105–114, 2017. doi: 10.1109/ICFEC.2017.8.

Jie Cao, Quan Zhang, and Weisong Shi. Challenges and opportunities in edge computing. SpringerBriefs Comput. Sci., pages 59–70, 2018. ISSN 21915776. doi: 10.1007/ 978-3-030-02083-5 5.

Djabir Abdeldjalil Chekired, Lyes Khoukhi, and Hussein T. Mouftah. Industrial IoT Data Scheduling based on Hierarchical Fog Computing: A key for Enabling Smart Factory. IEEE Trans. Ind. Informatics, 14(10):4590–4602, 2018. ISSN 15513203. doi: 10.1109/TII.2018.2843802.

Yung Chiao Chen, Yao Chung Chang, Chang Hsu Chen, Yu Shan Lin, Jiann Liang Chen, and Yu Yao Chang. Cloud-fog computing for information-centric Internet-of-Things applications. Proc. 2017 IEEE Int. Conf. Appl. Syst. Innov. Appl. Syst. Innov. Mod. Technol. ICASI 2017, pages 637–640, 2017. ISSN 978-1-5090-4897-7. doi: 10.1109/ICASI.2017.7988506.

D Chitra Devi and V Rhymend Uthariaraj. Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. Sci. world J., 2016, 2016.

Koustabh Dolui and Soumya Kanti Datta. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In 2017 Glob. Internet Things Summit, pages 1–6. IEEE, 2017.

Yifan Dong, Cheng Han, and Songtao Guo. Joint Optimization of Energy and QoE with Fairness in Cooperative Fog Computing System. In 2018 IEEE Int. Conf. Networking, Archit. Storage, pages 1–4. IEEE, IEEE, 2018. ISBN 9781538683675.

Chaker EL AMRANI and Hicham GIBET TANI. Smarter round robin scheduling algorithm for cloud computing and big data. J. Data Min. Digit. Humanit., 2018.

Dweepna Garg and Khushboo Trivedi. Fuzzy k-mean clustering in mapreduce on cloud based hadoop. In 2014 IEEE Int. Conf. Adv. Commun. Control Comput. Technol., pages 1607–1610. IEEE, 2014.

Lin Gu. K Meanses of Cloud Computing : MapReduce , DVM , and Windows Azure. (c):13–18, 2000.

Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. Softw. - Pr. Exp., 47 (9):1275–1296, 2017. ISSN 1097024X. doi: 10.1002/spe.2509.

Abdolreza Hatamlou, Salwani Abdullah, and Hossein Nezamabadi-Pour. A combined approach for clustering based on K-means and gravitational search algorithms. Swarm Evol. Comput., 6:47–52, 2012. ISSN 22106502. doi: 10.1016/j.swevo.2012.02.003. URL http://dx.doi.org/10.1016/j.swevo.2012.02.003.

Doan Hoang and Thanh Dat Dang. FBRC: Optimization of task scheduling in fog-based region and cloud. Proc. - 16th IEEE Int. Conf. Trust. Secur. Priv. Comput. Commun. 11th IEEE Int. Conf. Big Data Sci. Eng. 14th IEEE Int. Conf. Embed. Softw. Syst., pages 1109–1114, 2017. doi: 10.1109/Trustcom/BigDataSE/ICESS.2017.360.

Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. J. Netw. Comput. Appl., 98(September):27–42, 2017. ISSN 10958592. doi: 10.1016/j.jnca.2017.09.002. URL http://dx.doi.org/10.1016/j.jnca.2017.09.002.

Michaela Iorga, Larry Feldman, Robert Barton, Michael J Martin, Nedim Goren, and Charif Mahmoudi. Draft SP 800-191, The NIST Definition of Fog Computing. NIST Spec. Publ., 800(March), 2017. ISSN 00845612. doi: 10.6028/NIST.SP.800-145. URL https://csrc.nist.gov/publications/detail/sp/800-191/draft{%}0Ahttps://csrc.nist.gov/csrc/media/publications/sp/800-191/draft/documents/ sp800-191-draft.pdf.

Gopal Chandra Jana and Sudatta Banerjee. Enhancement of QoS for fog computing model aspect of robust resource management. 2017 Int. Conf. Intell. Comput. Instrum. Control Technol. ICICICT 2017, 2018-Janua:1462–1466, 2018. doi: 10.1109/ICICICT1.2017.8342785.

Sabihe Kabirzadeh, Dadmehr Rahbari, and Mohsen Nickray. A hyper heuristic algorithm for scheduling of fog networks. Conf. Open Innov. Assoc. Fruct, pages 148–155, 2018. ISSN 23057254. doi: 10.23919/FRUCT.2017.8250177.

Ruba A B U Khurma, Heba A L Harahsheh, and Ahmad Sharieh. TASK SCHEDULING ALGORITHM IN CLOUD COMPUTING BASED ON MODIFIED ROUND ROBIN ALGORITHM. 96(17):5869–5888, 2018.

Hlabishi I Kobo, Adnan M Abu-Mahfouz, and Gerhard P Hancke. A survey on software-defined wireless sensor networks: Challenges and design requirements. IEEE access, 5: 1872–1899, 2017.

Nir Kshetri. Cloud computing in sub-saharan Africa. IT Prof., 15(6):64–67, 2013. ISSN 15209202. doi: 10.1109/MITP.2013.92.

Aparna Kumari, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. Fog computing for Healthcare 4.0 environment: Opportunities and challenges. Comput. Electr. Eng., 72(September):1–13, 2018. ISSN 00457906. doi: 10.1016/j.compeleceng.2018.08.015. URL https://doi.org/10.1016/j.compeleceng.2018.08.015.

Lindong Liu, Deyu Qi, Naqin Zhou, and Yilin Wu. A Task Scheduling Algorithm Based on Classification Mining in Fog Computing Environment. Wirel. Commun. Mob. Comput., 2018, 2018a. ISSN 15308677. doi: 10.1155/2018/2102348.

Zhen Liu, Jiawei Zhang, Yanan Li, Lin Bai, and Yuefeng Ji. Joint Jobs Scheduling and Lightpath Provisioning in Fog Computing Micro Datacenter Networks. J. Opt. Commun. Netw., 10(7):B152, 2018b. ISSN 1943-0620. doi: 10.1364/JOCN.10.00B152.

URL https://www.osapublishing.org/abstract.cfm?URI=jocn-10-7-B152.

Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing: Focusing on mobile users at the edge. arXiv Prepr. arXiv1502.01815, 2015.

Somayya Madakam, R Ramaswamy, and Siddharth Tripathi. Journal of Computer and Communications. J. Comput. Commun., 3(May):164–173, 2015. ISSN 2327-5219. doi: 10.4236/jcc.2015.35021. URL https://file.scirp.org/pdf/JCC{_}2015052516013923.pdf.

Redowan Mahmud and Rajkumar Buyya. Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit. Fog Edge Comput. Princ. Parad., (April):1–35, 2018.

Long Mai, Nhu Ngoc Dao, and Minho Park. Real-time task assignment approach leveraging reinforcement learning with evolution strategies for long-term latency minimization in fog computing. Sensors (Switzerland), 18(9), 2018. ISSN 14248220. doi: 10.3390/s18092830.

Mithun Mukherjee, Lei Shu, and Di Wang. Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. IEEE Commun. Surv. Tutorials, (c):1–1, 2018. ISSN 1553-877X. doi: 10.1109/COMST.2018.2814571. URL http://ieeexplore.ieee.org/document/8314121/.

B V Natesha and Ram Mohana Reddy Guddeti. Heuristic-Based IoT Application Modules Placement in the Fog-Cloud Computing Environment. In 2018 IEEE/ACM Int. Conf. Util. Cloud Comput. Companion (UCC Companion), pages 24–25. IEEE, 2018.

OpenFog Consortium Architecture Working Group. OpenFog Reference Architecture for Fog Computing. OpenFog, (February):1–162, 2017. ISSN 2047-4954. doi: OPFRA001.020817.

Zhengyuan Pang, Lifeng Sun, Zhi Wang, Erfang Tian, and Shiqiang Yang. A Survey of Cloudlet Based Mobile Computing. Proc. - 2015 Int. Conf. Cloud Comput. Big Data, CCBD 2015, pages 268–275, 2016. doi: 10.1109/CCBD.2015.54.

Abelardo Pardo, Jesus Arias Fisteus, and Carlos Delgado Kloos. A distributed collaborative system for flexible learning content production and management. J. Res. Pr. Inf. Technol., 44(2):203–221, 2012. ISSN 1443458X. doi: 10.1002/spe.

Congduc Pham, Abdur Rahim, and Philippe Cousin. WAZIUP: A Low-Cost Infrastructure for Deploying IoT in Developing Countries. Lect. Notes Inst. Comput.

Sci. Soc. Telecommun. Eng. LNICST, 208:135–144, 2018. ISSN 18678211. doi: 10.1007/978-3-319-66742-3 13.

Xuan Qui Pham and Eui Nam Huh. Towards task scheduling in a cloud-fog computing system. 18th Asia-Pacific Netw. Oper. Manag. Symp. APNOMS 2016 Manag. Softwarized Infrastruct. - Proc., 2016. doi: 10.1109/APNOMS.2016.7737240.

Xuan Qui Pham, Eui Nam Huh, Kay Bierzynski, Antonio Escobar, Matthias Eberl, Gopal Chandra Jana, Sudatta Banerjee, Sabihe Kabirzadeh, Dadmehr Rahbari, Mohsen Nickray, Redowan Mahmud, Rajkumar Buyya, Koustabh Dolui, Soumya Kanti Datta, Mithun Mukherjee, Lei Shu, Di Wang, Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H. Luan, Hao Liang, Yang Yang, Shuang Zhao, Wuxiong Zhang, Yu Yijin Chen, Xiliang Luo, Jun Wang, Luxiu Yin, Juan Luo, Haibo Luo, Nan Wang, Blesson Varghese, Michail Matthaiou, Dimitrios S. Nikolopoulos, Doan Hoang, Thanh Dat Dang, Ashkan Yousefpour, Genya Ishigaki, Jason P. Jue, Daniele Santoro, Daniel Zozin, Daniele Pizzolli, Francesco De Pellegrini, Silvio Cretti, Dadmehr Rahbari, Mohsen Nickray, Indian J Sci, Qianyu Liu, Yunkai Wei, Supeng Leng, Yu Yijin Chen, Kevin Bachmann, Assistant Dr-Ing Stefan Schulte Mitwirkung, Olena Skarlat, Kevin Bachmann Stefan Schulte, Guenter I Klas, Tim Wauters, Bruno Volckaert, Filip De Turck, Shanhe Yi, Cheng Li, Qun Li, and Eugen Borcoci. Task Scheduling in Fog Enabled Internet of Things for Smart Cities. IEEE Internet Things J., PP(c):1–1, 2017. ISSN 2327-4662. doi: 10.1109/FMEC.2017.7946409. URL http://www.infosys.tuwien.ac.at/staff/sschulte/paper/Bachmann{_}Master.pdfhttp://ieeexplore.ieee.org/document/8331084/http://ieeexplore.ieee.org/document/8314121/.

Weijie Qi, Baoling Zhang, Bozhong Chen, and Jie Zhang. A user-based K-means clustering offloading algorithm for heterogeneous network. 2018 IEEE 8th Annu. Comput. Commun. Work. Conf. CCWC 2018, 2018-Janua(January 2019):307–312, 2018. doi: 10.1109/CCWC.2018.8301769.

Dadmehr Rahbari and Mohsen Nickray. Scheduling of Fog Networks with Optimized Knapsack by Symbiotic Organisms Search.

T Venkat Narayana Rao, Amer Khan, M Maschendra, and M Kiran Kumar. A Paradigm Shift from Cloud to Fog Computing. Ijcset, 5(11):385–389, 2015.

Farzad Samie. Resource Management for Edge Computing in Internet of Things (IoT). PhD Thesis, Karlsruher Inst. Technol., 2018.

Daniele Santoro, Daniel Zozin, Daniele Pizzolli, Francesco De Pellegrini, and Silvio Cretti. Foggy: A Platform for Workload Orchestration in a Fog Computing Environment. Proc.

Int. Conf. Cloud Comput. Technol. Sci. CloudCom, 2017-Decem:231–234, 2017. ISSN 23302186. doi: 10.1109/CloudCom.2017.62.

Jos´e Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. Fog computing: Enabling the management and orchestration of smart city applications in 5g networks. Entropy, 20(1):4, 2017.

Indian J Sci. A SURVEY ON RESOURCE MANAGEMENT FOR FOG-ENHANCED SERVICES AND Ameenabegum H Attar , 2 Ashok Sutagundar Department of Electronics and Communication Engineering , Basaveshwar Engineering College , Bagalkot. 17(2):138–141, 2018.

Anuj Sehgal, Vladislav Perelman, Siarhei Ku¨ryla, and Jurgen Sch¨onw¨alder. Management of resource constrained devices in the internet of things. IEEE Commun. Mag., 50(12): 144–149, 2012. ISSN 01636804. doi: 10.1109/MCOM.2012.6384464.

Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges. IEEE Internet Things J., 3(5):637–646, 2016. ISSN 23274662. doi: 10.1109/JIOT.2016.2579198.

Makhan Singh and Sarbjeet Singh. Information Dispersal Algorithms and Their Applications in Cloud Computing. 4(04):72–94, 2017. doi: 10.4018/978-1-5225-3029-9.ch004.

Pushpraj Singh, Vinod Singh, Anjani Pandey, and Round Robin. Analysis and Comparison of CPU Scheduling Algorithms. Int. J. Emerg. Technol. Adv. Eng., 4 (1):91–95, 2014.

Hicham Gibet Tani, Chaker El Amrani, Hicham Gibet Tani, Chaker El, Amrani Smarter, Round Robin, and Scheduling Algorithm. Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data To cite this version : HAL Id : hal-01443713 Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big. 2018.

John Thomson. Cloud Computing for Enterprise Architectures. (November), 2011. ISSN 0717-6163. doi: 10.1007/978-1-4471-2236-4. URL http://link.springer.com/10.1007/978-1-4471-2236-4.

Fernando Vargas Vargas. Cloudlet for the Internet-of- Things. 2016. URL https://kth.diva-portal.org/smash/get/diva2:956399/FULLTEXT01.pdf.

P.H. Vilela, J.J.P.C. Rodrigues, L.R. Vilela, M.M.E. Mahmoud, and P. Solic. A Critical Analysis of Healthcare Applications over Fog Computing Infrastructures. 2018 3rd Int. Conf. Smart Sustain. Technol. Split. 2018, pages 1–5, 2018.

Nan Wang, Blesson Varghese, Michail Matthaiou, and Dimitrios S Nikolopoulos. ENORM: A Framework For Edge NOde Resource Management. IEEE Trans. Serv. Comput., X(JANUARY):1–14, 2017. ISSN 19391374. doi: 10.1109/TSC.2017.2753775.

Yong Xiao and Marwan Krunz. QoE and power efficiency tradeoff for fog computing networks with fog node cooperation. Proc. - IEEE INFOCOM, 2017. ISSN 0743166X. doi: 10.1109/INFOCOM.2017.8057196.

Shanhe Yi, Zijiang Hao, Zhengrui Qin, and Qun Li. Fog computing: Platform and applications. In 2015 Third IEEE Work. Hot Top. Web Syst. Technol., pages 73–78. IEEE, 2015a.

Shanhe Yi, Cheng Li, and Qun Li. A Survey of Fog Computing : Concepts , Applications and Issues. 2015b.

Shanhe Yi, Zhengrui Qin, and Qun Li. Security and privacy issues of fog computing: A survey. In Int. Conf. Wirel. algorithms, Syst. Appl., pages 685–695. Springer, 2015c.

Luxiu Yin, Juan Luo, and Haibo Luo. Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacture. IEEE Trans. Ind. Informatics, PP(c):1, 2018. ISSN 15513203. doi: 10.1109/TII.2018.2851241.

Ashkan Yousefpour, Genya Ishigaki, and Jason P. Jue. Fog Computing: Towards Minimizing Delay in the Internet of Things. Proc. - 2017 IEEE 1st Int. Conf. Edge Comput. EDGE 2017, pages 17–24, 2017. doi: 10.1109/IEEE.EDGE.2017.12.

Decheng Zhang, Faisal Haider, Marc St-Hilaire, and Christian Makaya. Model and algorithms for the planning of fog computing networks. IEEE Internet Things J., 6(2):3873–3884, 2019. ISSN 23274662. doi: 10.1109/JIOT.2019.2892940.

Guowei Zhang, Fei Shen, Zening Liu, Yang Yang, Kunlun Wang, and Ming-Tuo Zhou. FEMTO: Fair and energy-minimized task offloading for fog-enabled IoT networks. IEEE Internet Things J., 2018a.

Haijun Zhang, Na Liu, Xiaoli Chu, Keping Long, Abdol-Hamid Aghvami, and Victor C M Leung. Network slicing based \uppercase{5G} and future mobile networks: mobility, resource management, and challenges. \uppercase{IEEE} Commun. Mag., 55(8):138–145, 2017.

Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu, and Wei Zhou. A Comparative Study of Containers and Virtual Machines in Big Data Environment. 2018b. ISSN 00946354. doi: 10.1109/CLOUD.2018.00030. URL http://arxiv.org/abs/1807.01842.

Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Yla-Jaaski. Folo: Latency and quality optimized task allocation in vehicular fog computing. IEEE Internet Things J., 6(3):4150–4161, 2019. ISSN 23274662. doi: 10.1109/JIOT.2018.2875520.

Appendices

A Raw Data from Simulations

The raw results are found on the https://github.com/Mzizio/Masters/tree/master repository