Sudan University of Science and Technology

College of Graduate Studies

An Energy Efficient -aware Fog - enabled Cloud of Things Model for Healthcare

نموذج طاقة فعال لسحابة أشياء مبني على تقنية الضباب (Fog) للرعاية الصحية

A thesis Submitted in partial fulfillment for the Requirements of the Degree of Doctor Philosophy in computer science

By

Mukhtar Mohamed Edris Mahmoud

Supervised by

Prof. Dr. Joel J. P. C. Rodrigues

August 2018

DEDICATION

To my Lovely parents, Wife, and Daughters.

ACKNOWLEDGEMENTS

All praise to Allah (s.w.t) the most Gracious and most Merciful, by whose grace and blessing this work has been completed. I would like to take this opportunity while relying on the instruction of the Prophet to the effect that “whoever does not thank people does not thank Allah” to express my thanks and gratitude to those who have contributed in one way or another through their advice, criticism, and support in strengthening the quality of this work. I would like to express my gratitude to those who have helped me in my pursuit for knowledge.

I would especially like to express my deep and sincere gratitude to my supervisor **Prof. Dr. Joel J. P. C. Rodrigues**, for hiscontinuous guidance and endless support throughout the length of this study. He has greatly helped me in a lot of ways I needed to go through this study. I am grateful to him for giving his wide knowledge, time and guidance to help me overcome the challenges in my study. I am also immensely grateful to **Prof Elzzedin M Osman** for his kind cooperation, as well as to all staff of Sudan University who extended their best cooperation during my study and their professionalism of tackling my personal obstacles.

My deepest thanks go to my parents, brothers, and sisters. Their influence made me realize the importance of education from a very early age. I also offer the deepest gratitude to my sweet hearts – my wife, my little girls (Daniah&Muzen) – for bearing my ignorance towards them during the journey of this study.

PUBLICATIONS

Most of the research work done during the PhD has been presented in the following papers:

1. **Towards energy-aware fog-enabled cloud of things for healthcare**

   Mukhtar M. E. Mahmoud,Joel J. P. C. Rodrigues, Saleem, K., Al-Muhtadi, J., Kumar, N. and Korotaev, V.

   Computers & Electrical Engineering, Elsevier, Vol. *67*, 2018, pp.58-69.

2. **Enabling Technologies on Cloud of Things for Healthcare**

   Mukhtar M. E. Mahmoud, Joel J. P. C. Rodrigues, Syed H. Ahmed, Sayed C. Shah, Jalal Al-Muhtadi, Valery Korotaev, Victor H. Albuquerque

   DOI: 10.1109/ACCESS.2018.2845399, IEEE Access,(Early Access)2018.

3. **A Critical Analysis of Healthcare Applications Over Fog Computing Infrastructures**

   Vilela, Pedro H., Joel JPC Rodrigues, Luciano R. Vilela, Mukhtar ME Mahmoud, and Petar Solic.

   In 2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech), pp. 1-5. IEEE, 2018.

ABSTRACT

The Internet-of-Things (IoT) represents the next groundbreaking change in information and communication technology (ICT) after the Internet. IoT is concerned with making everything connected and accessible through the Internet. However, IoT objects (things) are characterized by constrained computing and storage resources. Therefore, the Cloud of Things (CoT) paradigm that integrates the Cloud with IoT is proposed to meet the IoT requirements. This combination generates a new paradigm for pervasive and ubiquitous computing. In CoT, the IoT capabilities (e.g., sensing) are provisioned as services. Unfortunately, the two-tier CoT model is not efficient in the use cases sensitive to delays and energy consumption (e.g., in healthcare). Consequently, Fog Computing is proposed to support such IoT services and applications. This research analyses CoT architectures and platforms, as well as the implementation of CoT in the context of smart healthcare. Subsequently, the research explains some related issues of CoT, including the lack of standardization. Moreover, it focuses on energy efficiency with an in depth analysis of the most relevant proposals available in the literature. Furthermore, it proposes an energy-aware allocation algorithm for placing application modules (tasks) on Fog devices. Finally, the performance of the proposed strategy is evaluated in comparison with the default allocation and Cloud-only policies, using the iFogSim simulator. The proposed solution was observed to be more energy-efficient, saving approximately 2.72% of the energy compared to Cloud-only and approximately 1.6% of the energy compared to the Fog-default.

المستخلص

يمثل إنترنت الأشياء (Internet of Things) التغيير القادم والرائد في تكنولوجيا المعلومات والإتصالات (ICT) بعد الإنترنت. يهتم إنترنت الأشياء بجعل كل شئ (إنسان، حيوان، أجهزة، حساسات، ... إلخ) متصلا كما يمكن الوصول إليه عبر الإنترنت. إلا أن كائنات إنترنت الأشياء (Things) تتميز بموارد معالجة وتخزين محدودة. لذلك ظهر نموذج سحابة الأشياء (Cloud of Things) الذي يدمج الحوسبة السحابية مع إنترنت الأشياء لتلبية متطلبات إنترنت الأشياء كالتخزين والمعالجة. في سحابة الأشياء، يتم تقديم إمكانيات إنترنت الأشياء (مثل الإستشعار) كخدمات عند الطلب. لسوء الحظ، فإن نموذج سحابة الأشياء ثنائي الطبقات (Two-tier CoT) غير فعال في حالات الخدمات الحساسة للتأخير وإستهلاك الطاقة (مثل الرعاية الصحية لذوي الأمراض المزمنة). لهذا السبب، ظهر نموذج الـ (Fog Computing) لدعم مثل هذه الخدمات. يقوم هذا البحث بتوضيح وتحليل المفاهيم الأساسية لحوسبة الأشياء، كما يوضح تطبيق سحابة الأشياء في مجال الصحة الذكية. أيضا، يناقش البحث بعض القضايا في مجال سحابة الأشياء والتي ما زالت تمثل قضايا بحثية مفتوحة كفعالية الطاقة وإدارة وتحليل البيانات الكبيرة. علاوة على ذلك، يقترح البحث نموذج ثلاثي الطبقات بإضافة طبقة الـ Fog كطبقة وسيطة بين طبقتي إنترنت الأشياء والحوسبة السحابية، كما يقترح أيضا خوارزمية لتخصيص أو وضع المهام (الوحدات البرمجية) على أجهزة المعالجة الموجودة في طبقة الـ Fog بطريقة فعالة للطاقة. أخيرا، يتم تقييم أداء الخوارزمية مقارنة بالسياسات الإفتراضية المستخدمة في طبقة الـ Fog طبقة الحوسبة السحابية، بإستخدام برنامج المحاكاة iFogSim. من خلال النتائج، لوحظ أن الحل المقترح أكثر كفاءة في حفظ الطاقة، حيث يوفر حوالي 2.72% من الطاقة مقارنة بالحوسبة السحابية فقط، وحوالي 1.6% من الطاقة مقارنة بالسياسة الإفتراضية في طبقة الـ Fog، كما أنه يوفر حوالي 8% من الطاقة في طبقة الـ Fog لوحدها.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

API	Application Programming Interface
AAL	Ambient Assisted Living
CABAN	Cloud-Assisted Body Area Networks
CoT	Cloud of Things
CMN	Community Medical Network
DC	Data Center
D2D	Device-to-Device
DVFS	Dynamic Voltage and Frequency Scaling
e-Health	electronic Health
ECG	Electrocardiography
ICT	Information and Communication Technology
IaaS	Infrastructure as a Service
IoT	Internet of Things
IRR	Improved Round Robin
BSN	Body Sensor Network
BAN	Body Area Network
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
nDC	nano Data Center
OpenFog RA	OpenFog Reference Architecture
OLA	Opportunistic Large Array
WSN	Wireless Sensor Network
WBAN	Wireless Body Area Network
QoS	Quality of Service
PoE	Power-over-Ethernet
PaaS	Platform as a Service
RA	Reference Architecture
RFID	Radio Frequency Identification
SOA	Service Oriented Architecture
SLA	Service Level Agreement
SenaaS	Sensing as a Service

SaaS	Software as a Service
SPHERE	Sensor Platform for Healthcare in a Residential Environment
VCC	Virtual Cloud Carer
VM	Virtual Machine
VO	Virtual Objects
6LoWPAN	IPV6 over Low-Power Wireless Personal Area Networks

CHAPTER I

INTRODUCTION

1.1 Introduction

The advancements in information and communications technology (ICT) has in recent years led the healthcare community to progressively use such technologies to enhance the quality of existing service and to reduce their costs(Hans *et al.*, 2010; Avancha, Baxi and Kotz, 2012; Jara *et al.*, 2012; Mohammed *et al.*, 2014; Sebestyen *et al.*, 2014; Catarinucci *et al.*, 2015). Alvarez (Alvarez, 2002) defines e-Health as "e-Health is a consumer-centered model of health care where stakeholders collaborate, utilizing ICTs, including Internet technologies to manage health, arrange, deliver and account for care, and manage the health care system". E-Healthcare offers an excellent opportunity for patients to improve the quality of their lives by allowing them to carry on with their daily activities normally, while the physicians are monitoring them and providing them with consultation and health advice(Jara, Zamora-Izquierdo and Skarmeta, 2013; Lu, Lin and Shen, 2013).

Due to the rapid increasing of chronic diseases – particularly in developing countries – the use of ICT is crucial for early detection and prevention of these diseases besides reducing the expenditure on healthcare, which would protect healthcare budgets of these developing countries (Lu, Lin and Shen, 2013; Hamdi *et al.*, 2014; Hassanalieragh *et al.*, 2015). Community healthcare monitoring, for example, is very useful project in which an IoT-based network is established in a limited area or local community to promote healthcare services remotely to reduce the risks of chronic diseases(Riazul Islam *et al.*, 2015). Most of the proposed cloud-based IoT healthcare monitoring frameworks have three major components: data acquisition for using wearable sensors, data transmission which

is responsible for real-time sending the captured data to the data center of the healthcare organization in a secure manner, and cloud processing for data storage, analytics, and visualization(Hassanalieragh *et al.*, 2015). Generally, the e-Healthcare monitoring system is composed of: (i) Set of sensors, either smart or otherwise sensors, for capturing physiological parameters of the patients; (ii) Wireless Body Area Network (WBAN) based IoT communication to allow Machine-to-Machine (M2M) communication among Things or enabling physicians to remotely interact with medical server; (iii) Medical server on the Cloud for data storage, processing, and analytics; and (iv) Clinical stations refer to physicians (e.g., doctors) who have the ability to get information remotely from the medical server in the Cloud (Sawand *et al.*, 2015).

Nowadays, the research in this field is heading towards the integration of Cloud Computing and Internet of Things (Distefano, Merlino and Puliafito, 2012; Koubaa and Shakshuki, 2015). This amalgamation is referred to as Cloud of Things (CoT), which is a new paradigm that exploits the integration of two different and popular technologies – the Internet of Things and the Cloud – to promote Future Internet applications (Botta, De Donato, *et al.*, 2016). Although both IoT and Cloud are two different and independent technologies, there is a need to integrate them to complement each other and be able to support pervasive and ubiquitous computing (Rohokale, Prasad and Prasad, 2011; Suciu, Suciu and Fratu, 2013). Since the Things interconnected to the Internet are expected to reach 50 billion by 2020, there is a fast growing need to deal with massive amounts of data generated by these smart objects regarding storage and processing (Doukas and Maglogiannis, 2012; Al-Fuqaha *et al.*, 2015). CoT aims to reveal Things as a service through APIs, and make them available to other IoT applications (Kim and Kim, 2015; Díaz, Martín and Rubio, 2016b), which will enable those applications to exploit and deploy smart things to build smart integrated services without deploying their things as shown in Figure1. Reliable CoT-based services (particularly, for delay-sensitive services like e-Healthcare)

require energy efficient CoT architectures. Although many solutions have been proposed for energy efficient architectures, most of them have been made on IoT and Cloud separately (Al-Fuqaha *et al.*, 2015).

Highly delay-sensitive services such as healthcare services need reliable CoT architectures. Energy efficient CoT architectures can increase the Sensor Network lifetime, which improves the quality of the existing services(Chang, 2014).

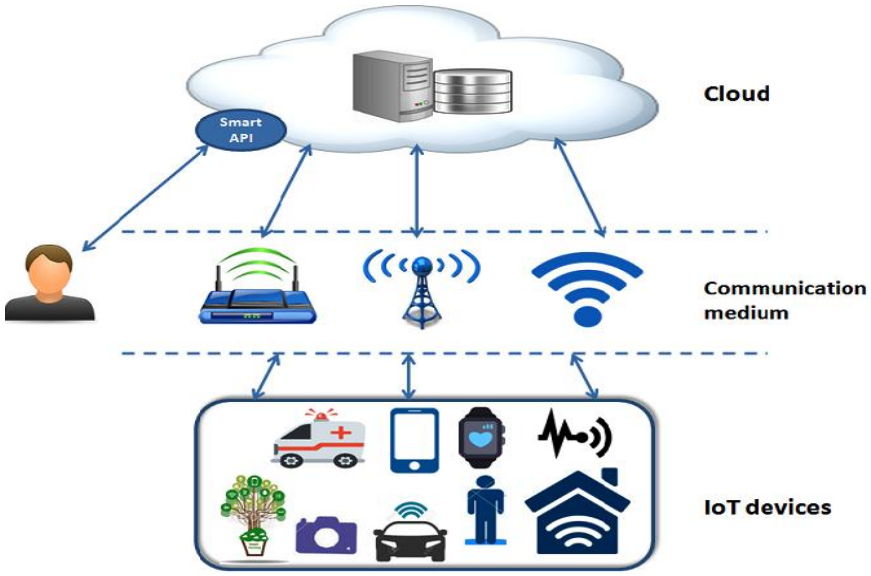

Figure 1.1.An illustration of the Cloud of Things concept, in which IoT objects are deployed and shared through smart APIs.

1.2 Background of Problem

Energy consumption plays a vital role when dealing with IoT-based healthcare monitoring. Therefore, many platforms and healthcare monitoring architecture are built in order to preserve better healthcare services in terms of energy consumption.

Vandana Milind Rohokale et al (Rohokale, Prasad and Prasad, 2011) proposed IoT-based cooperative approach for monitoring and controlling the essential human being's healthcare parameters (i.e. blood pressure, blood sugar, hemoglobin, etc) in rural areas. The approach is based on opportunistic large array (OLA) which represents a cluster of network of

nodes with active scattering mechanism for the received signals from the source. Due to the flexibility and scalability of OLA, the simulation results show that the cooperative approach is reliable and saving about 57% of energy. However, this approach didn't gain the benefits of integrating IoT with Cloud.

In (You, Liu and Tong, 2011), a community medical network (CMN) is proposed for local medicine and health system based on GSM/3G/WBAN infrastructures. The main objectives of CMN are supporting mobility, and reducing time and cost of diagnosis and treatment of diseases. CMN uses a gateway that utilizes smart relay for connecting WBAN with mobile network. The results show that the CMN is efficient in terms of economical and social. However, this architecture ignores the energy consumption issue.

An energy efficient architecture based on intelligent gateway is proposed in (Granados *et al.*, 2014) for pervasive healthcare. This architecture composed of: Power-over-Ethernet (PoE) enabled switch, cloud computing based platform for broadcasting and big data management, and web clients. The gateway receives commands that are proxies to sensors and actuators. After that, the gateway translates these commands to the convenient sensor application and network layer protocols. The results show that the architecture is efficient when evaluated in terms of sampling rate, latency, cost and convenience. In addition, the proposed gateway can serve as power/data source for wired sensors.

In (Benharref and Serhani, 2014), a framework for healthcare monitoring is proposed based on Cloud and service oriented architecture (SOA). This framework makes use of wearable biosensors for gathering vital data from patients. Then, these data are stored on the Cloud, and it will be accessed easily for authorized users. The experimental test bed on iOS and Android prove that suitable and the monitoring overhead is somewhat low even in

case of data in mobile. However, the framework has scalability problem and needs more components to ensure security and QoS-SLA guarantee.

Charalampos Doukas et al. (Doukas and Maglogiannis, 2012) introduce a platform based on IoT and Cloud for mobile and wearable healthcare sensors management. The proposed system composed of two parts: 1) sensors for capturing and submitting body signals, and 2) Cloud infrastructure for storing and managing the captured data. The major features of this platform architecture are scalability, interoperability, and light access. However, the proposed platform didn't consider the energy consumption and needs further work to address this issue.

1.3 Problem Statement and its Significance

In Cloud of Things (CoT), obtaining energy efficiency in both data processing and transmission is an important open issue. However, most of the proposed solutions emphasized on the Cloud and IoT, separately. Therefore, CoT require more efficient solutions (for obtaining energy efficiency in both data processing and transmission). Energy efficiency plays a vital role in preserving such an improved healthcare services. Thus, proposing and energy-aware CoT-based model for Healthcare is crucial for improving medical coverage. By doing so, many lives can be saved by providing timely diagnosis and medical advice for patients, especially in rural areas. Moreover, economical benefits for developing countries can be achieved by reducing financial costs.

1.4 Research Question/Hypothesis/Philosophy

1.4.1 Research Question

The main question that will be addressed in this research is how to preserve efficient healthcare monitoring based on Cloud of Things. Also, there are some additional questions such as follows:

1. Is the proposed model has more energy efficiency when compared with other relevant models or proposals?
2. Is the proposed model works properly and preserve acceptable performance in terms of energy consumption, latency, and network bandwidth?

1.4.2 Research Hypothesis

The proposed Cloud of Things based model can preserve efficient healthcare monitoring in terms of energy consumption and latency.

1.4.3 Research Philosophy

The philosophy of the proposed solution is based on exploiting the integration among Cloud Computing, Fog Computing, and Internet of Things in order to reduce the energy consumption. Then, proposing an energy-aware allocation algorithm for the placement of application modules (tasks) on Fog so as to obtain more optimized energy efficiency at Fog devices.

1.5 Research Objectives

The main objective of this research is to propose a new energy efficient CoT-based model for Healthcare (in terms of data processing and transmission). To reach this main objective, the following partial objectives were defined:

1. Review the state of the art of Cloud of Things, Cloud-based IoT healthcare services, applications, strategies, and mechanisms with focusing on energy efficiency

2. Performance assessment of a healthcare service that will be used to evaluate and validate the proposed cloud of things based healthcare model

3. Design and construction of a new cloud of things based model for healthcare

4. Performance evaluation and validation of the proposed model in comparison with other available solutions in the literature. The experiments will be performed through simulation.

1.6 Research Scope

This research is mainly focus on investigating and proposing an energy-aware Cloud of Things based model for healthcare monitoring so as to provision timely diagnosis and treatment for patients with diabetes disease. This model will be implemented in the Fog gateway in order to select the suitable fog device for allocating application modules on it. After that, the proposed model is compared with other relevant ones to evaluate its efficiency.

1.7 Research Methodology

First, the research investigates the state of the art of IoT-based Healthcare monitoring, by reviewing the most recent related contributions in order to determine the open issues and gaps that are not filled yet. Second, a new solution will be proposed to address the energy consumption issue. Third, the proposed solution will be experimented on use case for remote patient monitoring system of patients with diabetes disease. Finally, a comparison between the proposed solution and the most well known relevant solutions

will be made in terms of energy consumption, latency, and network bandwidth using simulation.

1.8 ExpectedContributions

The expected contributions can be described as follow:

1. Proposing an energy-aware Cloud of Things based model for Healthcare.
2. Publishing papers that add some related information to the body of knowledge.

1.9 Thesis Organization

This thesis is organized into six chapters that show as follow:

-Chapter 2, Literature Review: this chapter investigate and review the state-of-the-art of CoT and its role in providing efficient e-Healthcare monitoring services, especially regarding energy efficiency. An overview of the survey in the research areas is covered by this chapter. This chapter also discusses and analyzes the open issues in CoT with the focus on the proposed solutions in energy efficiency.

-Chapter 3, Methodology: this chapter describes the methodology used to achieve the objectives of this research. It also specifying the operational framework and discussing the proposed model and the allocation strategy to reduce energy consumption at Fog devices.

-Chapter 4, Simulation Tool and the Implementation: this chapter describes the simulation tool used and why we chosen it. It also describes the use case for evaluating the proposed model

-Chapter 5, Analysis of Results: this chapter covers the analysis and discussion of the obtained results.

-**Chapter 6**, Conclusion and Future Work: this chapter discusses and highlights the contributions and findings of the research work and presents suggestions and recommendations for future study.

CHAPTER II

LITERATURE REVIEW

2.1 Introduction

This chapter investigates and reviews the state-of-the-art of Cloud of Things (CoT) and its role in providing efficient e-Healthcare monitoring services, especially regarding energy efficiency. An overview of the survey in the research areas is covered by this chapter. This chapter also discusses and analyzes the open issues in CoT with the focus on the proposed solutions in energy efficiency.

2.2 Background

The Internet of Things (IoT) is a promising and innovative paradigm in the future Internet, which deals with connecting everything (i.e., physical and virtual objects) over the Internet with sensing/actuating functions for gathering data (Rao *et al.*, 2012; Riazul Islam *et al.*, 2015). These interconnected things (smart objects) have the ability to interact with each other to perform different tasks such as sharing information, and decision coordination in a self-configurable fashion without human intervention (i.e., machine-to-machine interaction) (Al-Fuqaha *et al.*, 2015; Li, Xu and Zhao, 2015; Pandya and Champaneria, 2015; Whitmore, Agarwal and Da Xu, 2015).

The term Internet of Things was coined by Kevin Ashton in 1999 (Khodadadi, A.V. and R., 2016), when he said: "*Internet of Things has the potential to change the world just as the Internet did, maybe even more so*". The advancements in technology made sensors smaller, cheaper and enable large scale deployment. Therefore, many sensors, that is billions, are currently deployed, and this number

will multiply rapidly in the near future. The data captured by these sensors are not useful without understanding it, so context-aware computing is necessary to solve this challenge and to promote the IoT paradigm (Perera *et al.*, 2014; Al-Fuqaha *et al.*, 2015). IoT can be applied in many domains such as industry, environment, and society. An essential component of IoT is a sensor network (SN) which is a network of interconnected sensor nodes using either wired or wireless technology. SN represents the backbone of the IoT, i.e., it does not exist without it (Perera *et al.*, 2014). However, IoT devices have constrained capabilities mainly in terms of storage, processing power, and energy efficiency (C. S. and N. K, 2015; Gia, Jiang, *et al.*, 2015; Díaz, Martín and Rubio, 2016a). In order to achieve energy efficiency, a lot of interconnected devices require techniques and algorithms for enhancing node sensing processing, and sink node communication (Prasad and Ieee, 2012).

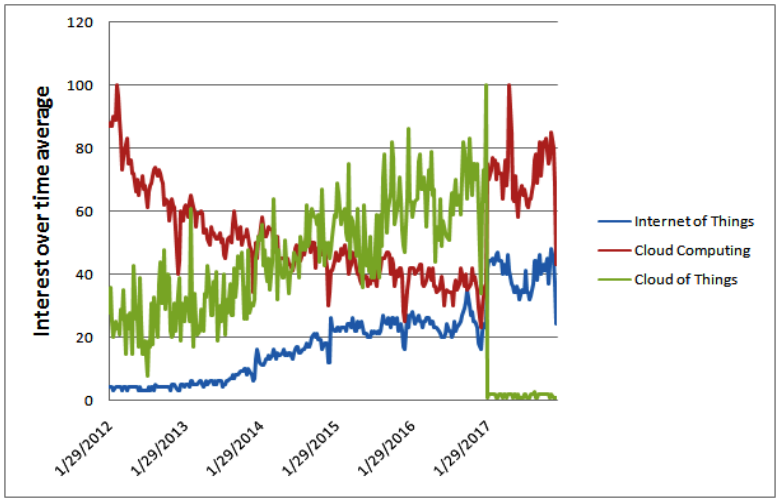

Figure 2.1.Current trend on Cloud, IoT, and CoT (source: Google Trends).

By 2020, a vast number of heterogeneous devices will be connected to our environment, which as a consequence will increase the volume of the produced data as well as the network traffic. Therefore, finding effective solutions to address the process of collecting, analyzing, managing, and storing for such huge and diverse quantities of data is crucial (Cavalcante *et al.*, 2016a).

In turn, the Cloud Computing (CC) is a promising paradigm for on-demand access to a shared set of resources such as networks, servers, storage, and services (Doukas and Maglogiannis, 2012; Botta *et al.*, 2014). Cloud Computing allows us to access those shared resources in an efficient and convenient manner (i.e., virtualization) without the need to maintain hardware resources (Pradhan, Behera and Ray, 2016). Cloud Computing appears as an urgent response to addressing the shortcomings of grid computing, such as the inability of resource access and its attachment to computing and data centers (Martinovic and Zoric, 2012). Cloud preserves ubiquitous computing capabilities, especially in terms of storage and processing power (Aazam *et al.*, 2014; Botta *et al.*, 2014; Al-Fuqaha *et al.*, 2015). From a complementary viewpoint, cloud computing can fulfill the main drawbacks of IoT.These drawbacks promote the trend towards integrating IoT with Cloud, which is known as Cloud of Things (CoT) (Díaz, Martín and Rubio, 2016a). Figure 2 explains the concerns related to the Cloud, IoT, and CoT. Despite the great benefits of Cloud, energy efficiency represents one of the important issues that needs appropriate solutions. Providing efficient scheduling schemes of virtual machines can significantly improve the energy efficiency of Cloud data centers when dealing with resource-intensive applications (Duan *et al.*, 2016).

2.3 Background studies and related Technologies

2.3.1 Cloud of Things

The Cloud of Things (CoT) is a new term that was coined by some researchers to refer to the integration between the cloud and the IoT (Aazam *et al.*, 2014). CoT paradigm aimed at bringing the IoT to the Cloud, in which, all IoT devices and capabilities can be accessed as a service through the Cloud (e.g. sensing as a service SenaaS). In CoT, Cloud acts as a middleware that makes the interaction between things and users/applications transparent (i.e., eliminates the complexity

which facilitates the development of applications that deal with smart objects) (Cavalcante *et al.*, 2016a). Cloud can benefit IoT with its virtually unlimited storage and computing resources, whereas IoT gives the Cloud the chance of extending its services to real world things (Babu, Lakshmi and Rao, 2015). Many efforts have been made to promote the trend toward this integration. Sensor-Cloud is one of the most important of these efforts, and is about blending sensors into the data center of the cloud and providing service-oriented access to sensor data and resources (Suciu, Suciu and Fratu, 2013). Many benefits can be tangible when exploiting the integration between Cloud and IoT such as follows.

- Efficient storage for IoT big data by exploiting the Cloud storage nature, i.e., On-demand, virtually unlimited and low-cost (Rohokale, Prasad and Prasad, 2011; Botta *et al.*, 2014).
- Regarding computation, the integration with Cloud enhances IoT processing and computation by adding more capabilities which are not allowed at the IoT end, and energy saving by enabling task offloading (Rohokale, Prasad and Prasad, 2011; Botta *et al.*, 2014; Fortino, Guerrieri, *et al.*, 2014; Babu, Lakshmi and Rao, 2015). In other words, the Cloud model satisfies the processing needs of IoT through its virtually unlimited processing and on-demand usage, which enables easier real-time analysis of IoT data.
- Cloud offers an efficient and low-cost solution to enable IoT to keep track and manage objects anywhere at any time without a need to communicate through expensive dedicated hardware. Moreover, it provides an efficient solution for managing the generated data of Things (Botta, De Donato, *et al.*, 2016).
- IoT has limitations in many areas such as scalability, interoperability and efficiency due to the high heterogeneity on its devices, technologies, and protocols. Cloud can facilitate the flow of IoT data collection and processing as well as ease the process of integration of new things while

reducing the cost of deployment and complex data processing (Botta, De Donato, *et al.*, 2016).

- In terms of scope, CoT promotes new smart services and applications that leverage the extension of Cloud through things, which opens new opportunities as well as new open issues (Babu, Lakshmi and Rao, 2015; Botta, De Donato, *et al.*, 2016).

The integration of Cloud with IoT generates a new promising paradigm in which all of the Cloud and IoT characteristics are absorbed as shown in Table 2.1.

Table 2.1: Features of IoT, Cloud, and CoT

IoT	**Cloud**	**CoT**
Pervasive in terms of resources placement from anywhere	Ubiquitous in terms of accessing resources from everywhere	Pervasive and ubiquitous in terms of placement and accessing of resources
Deals with real world objects (things)	Deals with virtual resources	Deals with real-world objects as well as virtual resources
Constrained capabilities in terms of storage and computing	Virtually unlimited storage and computing capabilities	Virtually unlimited storage and computing capabilities

2.3.2 Two-tier Cloud of Things

The Cloud of Things (CoT) (Botta, de Donato, *et al.*, 2016; Díaz, Martín and Rubio, 2016b) is a promising computing model in which IoT capabilities are preserved as on-demand services. In CoT, the Cloud and IoT complement each other. For instance, IoT can overcome resource constraints by taking advantage of virtually unlimited resources of the Cloud (Cavalcante *et al.*, 2016b). The Cloud can also augment its services (i.e., the scope of implementation) by interacting with things in the physical world. Furthermore, CoT simplifies the management of IoT devices/things by using the Cloud as a middleware between end-users/applications and things, reducing complexity and hence helping promote the development of smart services, such as pervasive healthcare.

In general, the two-tier CoT model (Li *et al.*, 2017) includes the IoT-tier (the Things-tier) and the Cloud-tier, as shown in Figure3.2. On the one hand, the IoT-tier comprises a plethora of physical things equipped with sensing/actuating functions. The IoT-tier contains a set of protocols for organizing things that share the same geographical region or behavior into clusters, as well as providing appropriate access (direct or indirect) to such things (e.g., mobile things). On the other hand, the Cloud-tier is responsible for storing and processing big data generated by the IoT-tier due to the extensive storage capacity and computing capability of the former. Various challenges have arisen in the currently existing two-tier CoT model, such as inefficient use of computing resources, unnecessary data redundancy, unpredictable latency, a lack of mobility support and energy consumption. Thus, the three-tier CoT model is introduced, in which Fog computing acts as an intermediate layer between the IoT and Cloud environments (Li *et al.*, 2017).

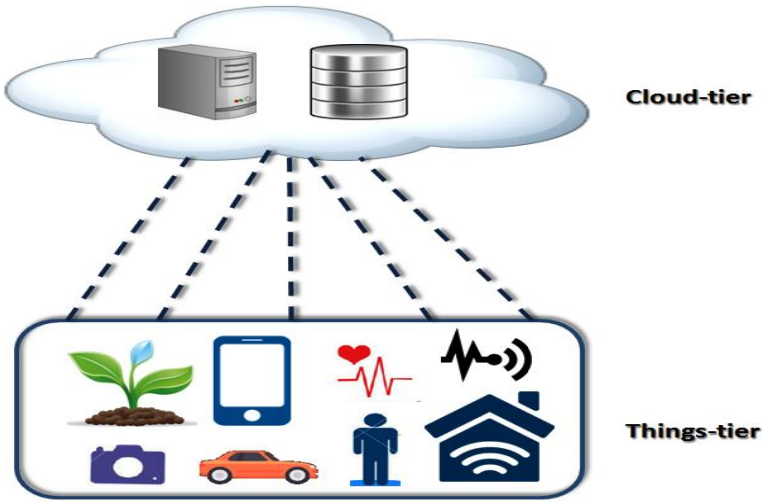

Figure 2.2. An illustration of the two-tier CoT model, in which IoT data of different types and volumes are handled by the Cloud (this is impractical).

2.3.3 Fog Computing

Due to the rapid growth of IoT services, the sole reliance on the Cloud is not adequate for meeting the requirements of such services, e.g., scalability and latency. To this effect, the Fog computing paradigm is introduced to enable a collaboration with the Cloud to fulfill the needs of IoT services (Chiang and Zhang, 2016). While being similar to the Cloud, the Fog provides storage and computing resources at the network edge rather than the core. Such proximity of Fog to end users and IoT devices (i.e., being closer to data consumers and producers) represents the proper way to provision IoT services that are computationally intensive or have real-time requirements. Fog resources (called Fog-nodes) can be constrained-resource devices (e.g., end devices) or powerful-resource devices (e.g., Cloudlets) (Shi, Ding, Wang, Roman, *et al.*, 2015).

Despite the benefits of the Fog paradigm, several questions need to be addressed, such as efficient management of Fog resources and the optimal placement of IoT services/applications on Fog devices. Few existing studies attempt addressing such questions individually. One of the most significant contributions that will influence the Fog significantly is the OpenFog Reference Architecture (OpenFog RA), proposed by the OpenFog Consortium (Consortium and Working, 2017). OpenFog RA aims to support the Fog community (including businesses, developers, etc.) with a set of rules and guidance on various aspects, such as scalability and security. Following such guidelines, efficient and robust Fog applications can be created according to the desired objectives.

2.3.4 Fog-enabled Cloud of Things

As mentioned above, the two-tier CoT model fails to enable IoT services that require predictable latency. Moreover, extensive communications with the Cloud consume network bandwidth and energy, in addition to increasing the load on a Cloud datacenter. Integrating Fog as a Middleware layer with CoT can successfully produce a new efficient architecture that satisfies the requirements of IoT services. In a Fog-enabled CoT model (see Fig. 3), the Fog and the Cloud operate interchangeably to execute service tasks according to whether the demanded response is delay-sensitive (Shi, Ding, Wang, Roman, *et al.*, 2015; Li *et al.*, 2017). Instead of providing real-time task execution and temporary storage, Fog can eliminate the redundant data and send only the filtered data to the Cloud for sophisticated analysis or permanent storage (Shi, Ding, Wang, Roman, *et al.*, 2015). To this end, smart gateways can choose to execute requests locally at the edge or forward them to the Cloud after filtering. Data filtering by the Fog reduces the amount of data transmitted, conserving energy and network bandwidth.

Several studies (Deng *et al.*, 2016; V. B. C. Souza *et al.*, 2016; V. B. Souza *et al.*, 2016; Xuan-Qui Pham and Eui-Nam Huh, 2016) investigated the interplay between the Fog and the Cloud. Some of the cited proposals presented new approaches to creating efficient solutions for scheduling and allocation of tasks and Fog resources, while the others analyze such interplay theoretically.

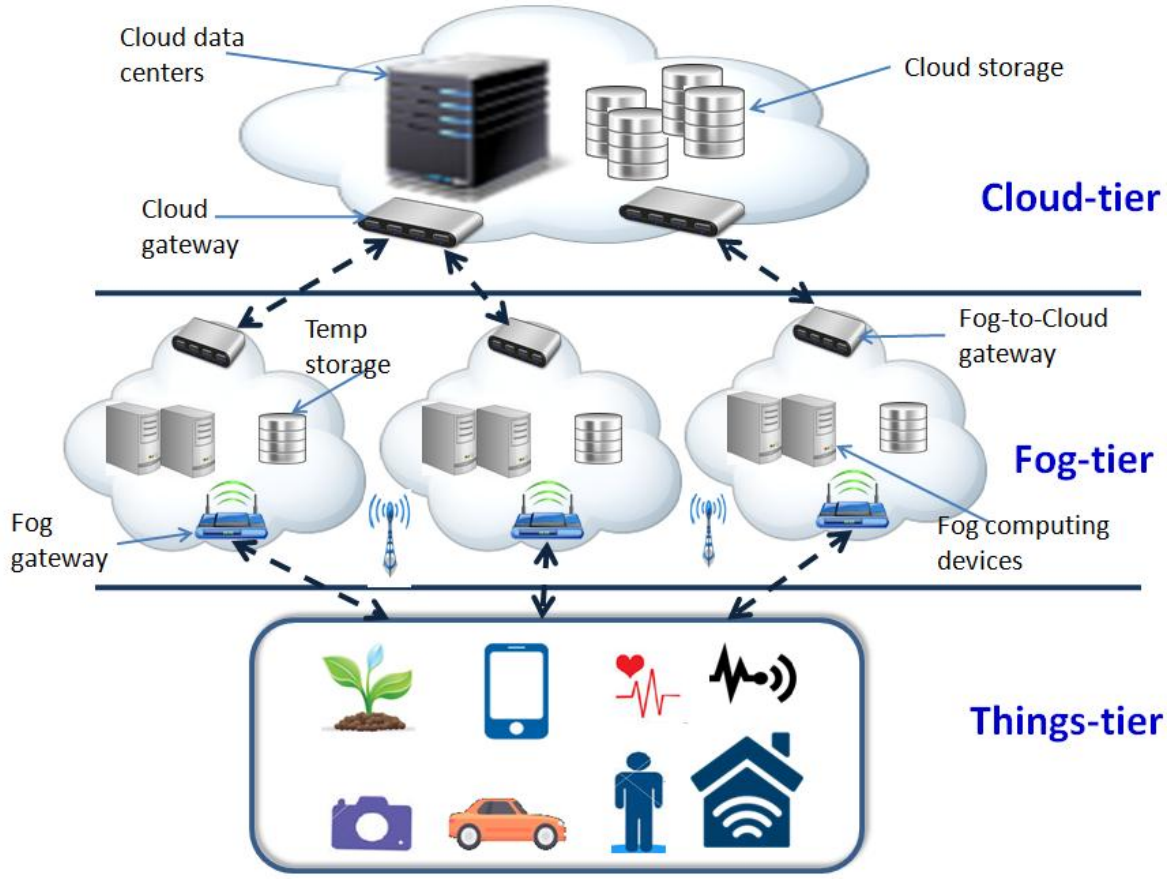

Figure 2.3.The three-tier CoT model, in which IoT data are handled by the proper Fog instance or forwarded to the Cloud after preprocessing operations.

2.3.5 Cloud of Things Architectures

The combination of IoT and Cloud (CoT) represent the ongoing trend for the next generation of IoT smart services. The data collected through IoT smart objects will be processed and analyzed on Cloud data centers to produce valuable information. However, the available Cloud architecture requires enhancements to be more efficient and convenient for IoT real-time services in terms of energy consumption and end-to-end delays. For this reason, Cloud architectures going to be distributed closer to the network edge (i.e., fog nodes, micro-cloud, and Cloudlets). Based on these distributed architectures, CoT networks can be more efficient and flexible regarding resource allocation, mobility support, low latency, reliability, and scalability (Barcelo *et al.*, 2016).

Several contributions introduce new CoT-based architectures that focus on gaining the benefits of integrating the Cloud with the IoT as well as addressing the major challenges that are arising such as data transmission (Cavalcante *et al.*, 2016a). This subsection summarizes the more relevant of these architectures.

L. Belli et al. (Belli *et al.*, 2015) introduced listener-based Graph Cloud architecture that intended to manage IoT Big Stream applications such as e-Health and Smart Cities. This architecture aimed at reducing data dispatching latency to consumers and enhancing resource allocation. The architecture (see Figure 3) composed of (i) Acquisition module, which is responsible for collecting raw data from the IoT objects and makes them available for other architecture blocks; (ii) Normalization Module, which normalizes the incoming data in a convenient format for processing; (iii) Graph Framework, which is a set of listeners represented by a node in the graph; (iv) Application Register for recording the interests of the listeners.

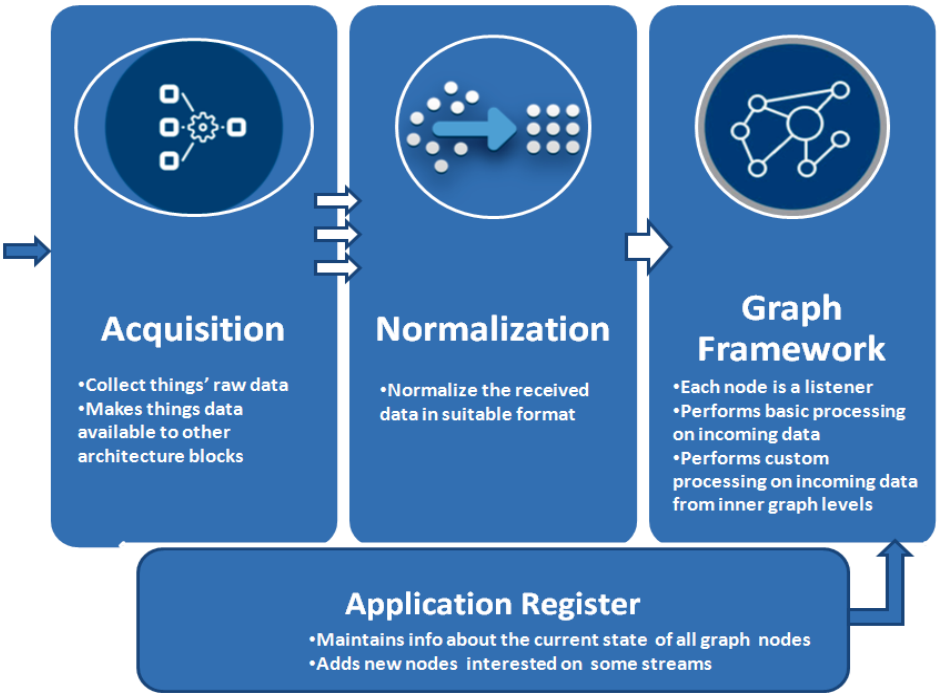

Figure 2.2.The Listener-based Graph Cloud Architecture Concept.

The architecture exploits the consumer-oriented data flow for data retrieval. The listener or consumer can determine the type of the incoming data (i.e., from Cloud service) either in a raw or processed format according to its registered interest. Furthermore, Cloud services can work as extra listeners that can be consumed by other end-users. The results show that the architecture is cost-effective for Cloud services regarding data dispatching latency and resource allocation.

In (Zhou *et al.*, 2013), an online platform architecture is proposed known as CloudThings. The architecture facilitates the development, deployment, operation, and composition of IoT applications by adapting the three Cloud models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) as shown in Figure 4. These adaptations, enable users to operate IoT-based applications on Cloud hardware (through CloudThings IaaS), ease the process of application development and decrease the expenditures of management and maintenance (through CloudThing PaaS), and facilitate storing, sharing, and managing things and events (through CloudThings SaaS).

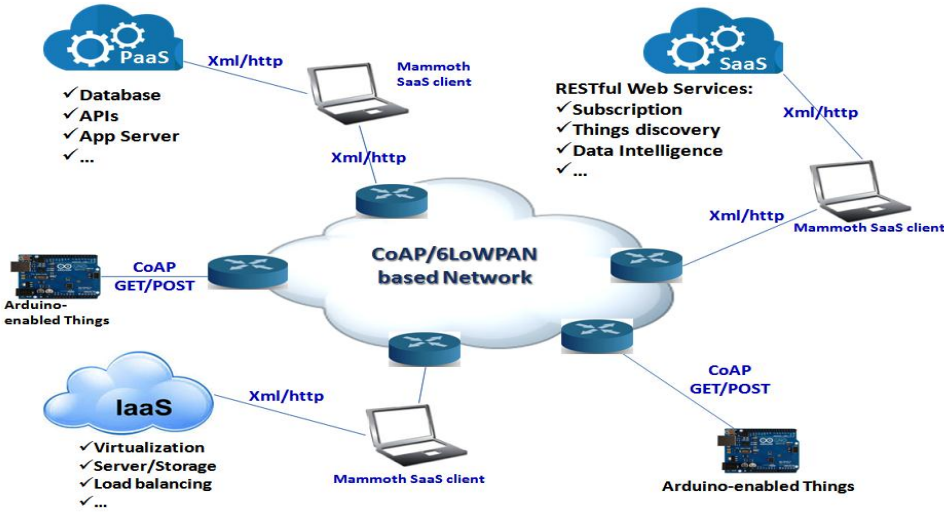

Figure 2.3.Illustration of the CloudThings architecture concept.

The authors of (Hou *et al.*, 2016) proposed an IoT Cloud architecture which combined with HTTP and MQTT servers to provide services to end-users and ensure real-time communication of a lot of connected devices respectively. The architecture is composed of IoT infrastructure, IoT Cloud for virtualizing IoT infrastructure as shown in Figure 5. The experimental results show that the architecture improves the performance in terms of transmission latency.

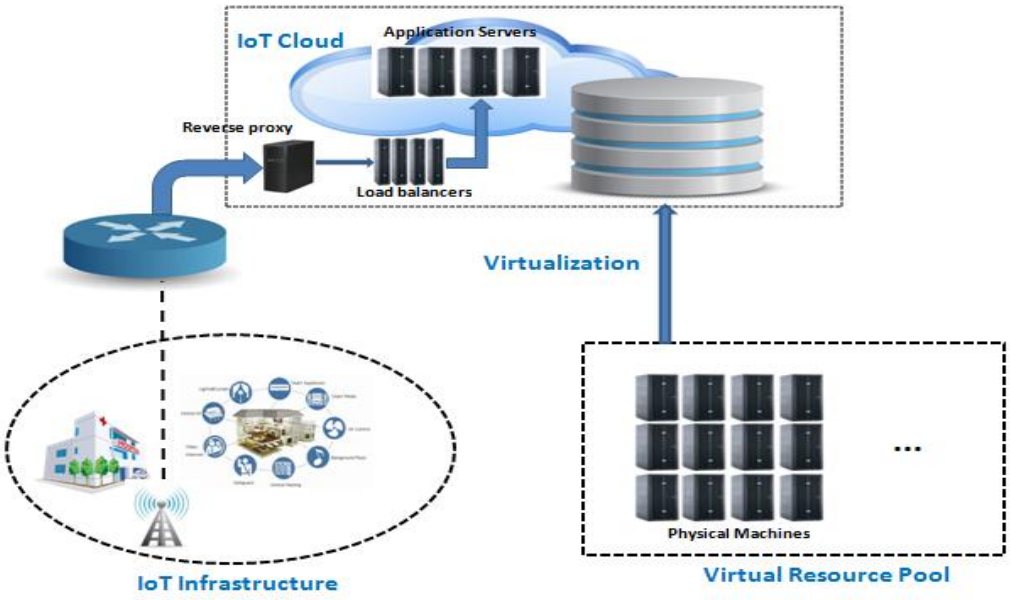

Figure 2.4.Illustration of an IoT cloud architecture.

2.3.6 Cloud of Things Platforms

The literature shows various efforts that have been made in order to develop a platform architecture that deals with the new Cloud of Things paradigm. These platforms, open sources or proprietary, are concerned about addressing heterogeneity issues related to both Cloud and IoT by implementing two middleware: the one on Cloud side and the other on the Things side, in addition to offering an API for interaction with applications. In this subsection, we review the most common of these platforms.

- **IoTCloud** (Botta, De Donato, *et al.*, 2016) is an open source platform for integrating IoT objects with the Cloud as well as offering an API interface for applications to interact with the data of IoT objects.
- **OpenIoT** (Wang and Wu, 2009; Belli *et al.*, 2015; Botta, De Donato, *et al.*, 2016; Díaz, Martín and Rubio, 2016b) is also an open source platform that acts as middleware for deploying and managing Cloud-IoT infrastructures. OpenIoT is about providing efficient organization of data collection and transmission to the Cloud regarding mobility and energy consumption. Furthermore, it facilitates the process of handling mobile sensors and related quality of service factors. Semantic interoperability is one of the key features that distinguishes OpenIoT from other CoT platforms (Soldatos *et al.*, 2015).
- **NimBits** (Zhou *et al.*, 2013; Botta, De Donato, *et al.*, 2016)is an open source platform based on a Cloud architecture that helps users deal with sensor data (i.e., record or share data) as well as enabling connection among things based on data points. With NimBits, compression and calculations can be achieved on data received from IoT devices using built-in mechanisms.

A comparison between the most relevant CoT platforms is summarized in Table 2.2.

Table 2.2: Features And Technology Used In The Selected Cot Platforms

Platform	Integration	Connection and data collection methods	Security techniques	Analytics types	Energy efficient?	Open source?
OpenIoT	Through REST API	X-GSN (extension of Global Sensor Networks)	oAuth 2.0	Not determined	Yes (especially in data collection)	Yes
Xively	Through REST API	MQTT, Sockets, WebSockets, and HTTP(S)	Link encryption using SSL/TLS	Not determined	No	No
NimBits	Through REST API	RESTful interfaces	oAuth 2.0 and keys	N/A	No	Yes
thingSpeak	Through REST API	HTTP and ZigBee	API keys	Allow analytics with MATLAB	No	Yes
CloudPlugs	Through REST API	PlugNet, MQTT, WebSocket. (ZigBee and Bluetooth	end-to-end security using PlugNet and	Allow analytics but unknown	Yes (reduce the communication)	No

		4 and Wi-Fi for local connections)	SSL,			
ParStream	UDX-based API	MQTT	Unknown	Real-time and Batch analytics	Yes (uses highly parallelized hardware architecture and compressed mode)	No
EVRYTHNG	Through REST API	MQTT, CoAP, and WebSockets	Link encryption using TLS, and oAuth 2.0, token-based API keys for services interaction	Real-time analytics	Yes (through THNGHUB gateway which reduces communication latency)	No
ThingWrox	Through REST API	Two-way non-polled communications through REST API or MQTT	end-to-end security, roles for permissions	Real-time anomaly detection and predictive analytics	Unknown	No

2.4 Cloud of Things in Healthcare

The convergence of Cloud Computing (CC) and the Internet of Things (IoT) have significantly changed the information technology industry. On the one hand, Cloud computing has helped in constructing efficient applications regarding scalability, virtualization, reliability and cost expenses. On the other hand, IoT with its innovative elements such as RFID (Radio Frequency Identification) and sensors could be successfully helps in realizing the world objects to achieve pervasive monitoring and management in a scalable region. Currently, Cloud and IoT are extensively applied in a large number of information technology applications. However, the ongoing research in healthcare toward the combination of Cloud and IoT does not meet the requirements. Bringing this integration in the context of healthcare can significantly contribute to building efficient healthcare applications for managing and monitoring hospitals and patients in an efficient manner regarding resource sharing and cost expenditures. Based on Cloud and IoT, remote healthcare monitoring and management information services can successfully provide early detection and treatment of chronic diseases that have a significant impact on people's health. That is, IoT body sensors (i.e., implantable or wearable) gather

the required information from a person, and then the information can be analyzed and processed in Cloud (Luo and Ren, 2016).

The use of CoT in healthcare domain offers new opportunities to medical IT infrastructure, and can enhance healthcare services (Catarinucci *et al.*, 2015). Moreover, CoT improves the healthcare processes and the quality of the actual healthcare services by simplifying the process of gathering patients' vital data and delivering them to a medical center on Cloud for storage and processing purposes (Botta, De Donato, *et al.*, 2016). In other words, the use of CoT in Body Sensor Network (BSN)-based healthcare helps in the process of storing gathered data, as well as processing and analyzing them in a scalable fashion (Fortino *et al.*, 2013). With CoT, healthcare sensors can be managed efficiently in a transparent manner as well as make any dealing with delay-sensitive healthcare services more efficient (Aazam and Huh, 2014). To achieve efficient healthcare services regarding delay-sensitive and energy consumption, fog computing can play a vital role by lowering the burden on Cloud as well as acting as a local storage for IoT devices and its ability to do some processing on the data (Sarkar, Chatterjee and Misra, 2015; Shi, Ding, Wang, Eduardo Roman, *et al.*, 2015; Ramalho *et al.*, 2016).

Because patients' data are sensitive, regulations do not allow them to be processed outside the healthcare organization (Kraemer *et al.*, 2017). Thus, Fog Computing is required to fill this gap by bringing processing capabilities closer to the healthcare providers (e.g., hospital). By doing this, a lot of benefits can be obtained, such as reduced latency and reduced energy consumption as well as improved bandwidth usage and data privacy. By using Fog-enabled CoT, the collected sensory data can be processed and analyzed in the local gateway (smart gateway) whereas physicians are able to access the results through the Cloud remotely. Following this approach reduces data transmission and execution times as well as saves energy. The types of healthcare device used depend on the application deployment scenario. For instance, in the mobile patient monitoring

deployment scenario, Smartphones can be used as a WPAN gateway for enabling direct connection to the WAN via cellular networks. In this scenario, the sensor devices, i.e., wearable or implantable, communicate with the WPAN through the gateway (i.e., the Smartphone). The connectivity of healthcare devices with the Cloud also depends on the healthcare application deployment scenario. For example, some scenarios may connect directly to the WLAN via Wi-Fi communication while in others it may connected using WPAN via Bluetooth (Kraemer *et al.*, 2017).

Besides the recognition and detection of patients' state changes, smartphone-based gateways can successfully offer transparent communication among IoT devices and the Internet as well as IoT devices management. For instance, work (Aloi *et al.*, 2017) proposed a high-level design of a smartphone-centric opportunistic gateway architecture to support flexible and transparent interoperability.

Due to the increased rate of chronic diseases in both developed and developing countries, the concern about developing healthcare projects based on ICT technologies for providing healthcare service has grown. For instance, Virtual Cloud Carer (VCC) [52] is a CoT-based project funded by Spanish National R&D intended to provide innovative healthcare services for dependent and elderly people with chronic diseases. By using CoT, the VCC project aims to gain social and technological objectives that enhance the quality of the new services being offered to the elderly. These objectives range from creating platform architecture – that is responsible for gathering physiological parameters of the elderly from anywhere and deploy them to the Cloud for storage and processing purposes – to helping the elderly do their physical training exercises and to support caregivers to track and monitor the elderly remotely in an efficient manner.

In (Luo and Ren, 2016), another CoT-based architecture model for remote monitoring and managing health was proposed. The proposed architecture aimed to provide effective healthcare monitoring and management along with improved performance and energy consumption. To reach this goal, the architecture proposed an algorithm (PSOSAA) based on the combination of two algorithms: the particle swarm optimization (PSO) and simulated annealing (SA). The simulation results showed the efficiency of the proposed algorithm in terms of energy efficiency and performance when compared with PSO and SA applied separately.

The authors of (Gia, Jiang, *et al.*, 2015) proposed an improved CoT-based healthcare monitoring system. The proposed system is based on a smart gateway which takes advantage of Fog Computing. It is clear that Fog in association with Cloud can improve the IoT-based applications/services regarding latency, location awareness, interoperability, and scalability. The proposed system is experimented on an ECG signals and the results reveal the role of Fog in providing an efficient healthcare monitoring system, in terms of bandwidth and low-latency by moving most of the processes to the network edge.

Although BSNs are intended to monitor the human body activities and capture vital signs, they cannot efficiently accomplish all these tasks due to their constrained resources. To address this, new infrastructures that integrate between BSNs and the Cloud are proposed and referred to as Cloud-Assisted Body Area Networks (CABAN) (Fortino, Di Fatta, *et al.*, 2014). With CABAN, BSN tasks can be achieved efficiently in terms of interoperability, scalability (in processing, data collection, and data storage), and ubiquitous access to resources. For instance, work (Fortino, Parisi, *et al.*, 2014), proposed a novel general-purpose system architecture, named BodyCloud, that exploits the combination of BSN and the Cloud to provision BSN-based applications as services such as storage,

processing, and management of sensor data streams. To achieve these goals, BodyCloud uses four decentralized parts: (i) Body: uses mobile devices to send the data captured, by wearable sensors, to the Cloud. (ii) Cloud: the core part which is responsible for providing a full support for particular applications from data collection to visualization.(iii) Viewer which enables advanced visualization of analysis results in a Web browser. (iv) Analyst: helps in developing BodyCloud-based applications. With BodyCloud, fast prototyping scalable, customizable, and interoperable Cloud-enabled BSN applications can be accomplished.

Also, in (Gravina *et al.*, 2017), a framework, named Activity-aaService, built on top of the BodyCloud architecture (Fortino, Parisi, *et al.*, 2014) is proposed to enable online/offline monitoring and recognition of individuals or communities activity (even in mobility). By taking the advantages of BodyCloud, Activity-aaService can allow a flexible creation of rapid prototyping activity-assisted applications. The evaluation of Activity-aaService is taken to measure the performance of high- and low-profile BSN coordinator devices in the Body-side only of BodyCloud platform. The results showed the efficiency of high-profile devices in terms of interoperability and reliability, while low-profile devices are not appropriate in the case of computation-intensive tasks.

SPHERE (Sensor Platform for Healthcare in a Residential Environment) Project (Zhu *et al.*, 2015) is a healthcare platform designed for monitoring and tracking patients at home (AAL). It improves healthcare detection and management through the use of valuable datasets that were extracted from the data gathered by the sensors. The SPHERE project architecture employs a cluster-based approach that is composed by (*i*) Ambient sensor network for gathering data of the patient, (*ii*) Home gateway for collecting data gathered by each sensor cluster besides supporting time synchronization and data privacy, and (*iii*) SPHERE data hub for storing the collected data in the home gateway, in addition, to supporting analytics. SPHERE project concerns about supporting the

healthcare community with a rich dataset that supports in enhancing the field of healthcare.

Most of the current efforts in healthcare (Fratu *et al.*, 2015; Ahmad *et al.*, 2016; Andriopoulou, Dagiuklas and Orphanoudakis, 2017), try to integrate Fog, IoT, and Cloud together and to gain the advantages of Fog such as low-latency and mobility support to improve the performance of the actual healthcare services. The combination of these technologies creates more efficient healthcare system architecture as explained in Figure 6.

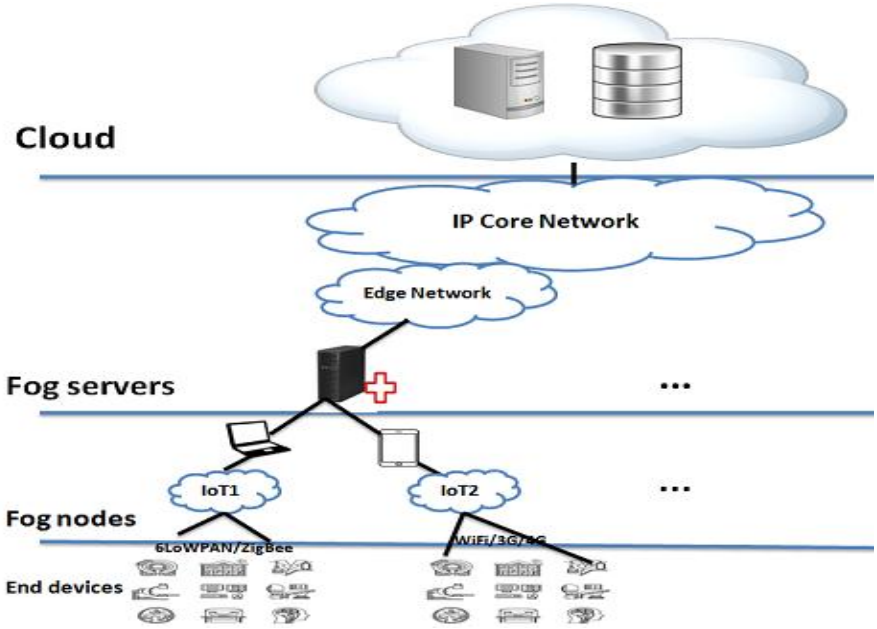

Figure 2.5.Illustration of IoT, Fog, and Cloud integration architecture.

2.5 Energy Efficiency Proposals

Energy efficiency represents one of the key issues that affect IoT services availability, reliability, and quality of service (QoS). This section summarizes the proposed solutions regarding energy efficiency.

Riccardo Petrolo et al. (Petrolo *et al.*, 2016) introduced a gateway architecture for Cloud of Things based on lightweight virtualization technologies. The gateway

can manage Things and works as a middleware between real-time data and consumer applications. Furthermore, the gateway uses prediction algorithms on data production to eliminate unnecessary communications between the gateway and Things, which reduces energy consumption. The results show that leveraging the combination of these technologies enhances service deployment, service management, and resource allocation. However, this gateway still needs further adoptions in prediction algorithms on data production to decrease the communication between the gateway and Things.

In (Chang, 2014), an energy-efficient scheme based upon distributed cluster computing is presented. By using a suitable routing structure, the scheme improves energy efficiency and increase the network lifetime by decreasing data transmission distance amongst sensor nodes (things). Typically, the operation has two phases, the setup and the steady-state. The setup phase is responsible for creating the cluster routing structure and selection of one node to act as the cluster head node (CH) according to the calculated center of gravity among the cluster nodes. In the steady-state phase, the CH node compresses the received data, i.e., from non-CH nodes, into a unique signal by performing signal processing. Subsequently, i.e., after data compression, the CH node sends the signal to the base station (BS). Obviously, compressing the data in the CH node reduces the amount of information transmitted. The scheme repeats these two phases for each round. The simulation results show the suitability of the scheme in gaining acceptable performance in terms of energy efficiency and network lifetime in wireless sensor networks and large-scale IoT-based systems. However, the main drawback of this scheme is the additional overhead in cluster head node selection.

An energy-efficient architecture for IoT is proposed in (Kaur and Sood, 2015). By using a sleep mode, the architecture reduces the energy consumption based on

switching sensors to sleep mode according to three cases: i) the necessity of sensing a targeted environment in a specific interval of time; ii) when the coverage area size exceeds the battery power, and iii) the battery level situation. Accordingly, whenever sensors switched to sleep mode, all the allocated cloud resources related to them are re-provisioned. Principally, the architecture design focuses on improving energy efficiency along the processes of sensing and transmission, extracting meaningful information from sensed data, and using the extracted data in a specific field. The results show that the proposed architecture is energy-efficient, scalable, and has proper performance when compared with other relevant schemes.

The authors of (Mohsen Nia *et al.*, 2015) introduced an energy-efficient scheme for long-term continuous personal health monitoring. Initially, the authors specified the requirements of the process of continuous personal health monitoring, by determining a sample resolution on each sensor. Subsequently, they proposed a CS-based (Compressive Sampling) scheme that composes of data collection, data transmission, and storage mechanisms. The results show the energy efficiency of the scheme. However, the scheme does not take into account the advantage of efficient storage and processing of the Cloud.

In (Mangali and Kota, 2015), the authors proposed a cluster-based scheme for e-health monitoring. This scheme divided the Whole wireless sensor network into Clusters including cluster-head and body-head nodes for routing data amongst them. Moreover, particular tasks could be assigned to cluster heads and to body heads, which significantly improve the efficiency of the whole monitoring system regarding scalability and energy efficiency. The results of the mathematical analysis show the energy efficiency of the scheme besides increasing the network lifetime. However, the scheme focuses only on lowering energy consumption in

the WSN part and ignores the Cloud part which affects energy consumption significantly.

The authors of (Granados *et al.,* 2014) proposed an IoT healthcare architecture based on a smart gateway that uses the Power over Ethernet (PoE) standard. By taking advantage of the PoE cable, the smart gateway can effectively transmit the data as well as supply IoT-enabled devices with power. Furthermore, the gateway can connect IoT medical sensors with the Cloud and process the gathered health data, which reduces the burden on IoT sensors. The results show that the gateway is an energy-efficient and cost-effective solution for healthcare. However, the gateway is efficient in the case of wired sensors and fixed scenarios, therefore, it needs further modifications to be suitable for wireless sensors and unfixed scenarios.

In (Gia, Thanigaivelan, *et al.*, 2015), the authors discussed the role of effective customization of 6LoWPAN (IPV6 over low power wireless area networks) in gaining energy efficiency and reliable IoT-based e-Health applications. With this architecture, patients' physiological data, such as ECG signals can be monitored, analyzed and transmitted in an efficient manner through the customized 6LoWPAN. Typically, the architecture composed of the customized 6LoWPAN network intended to deal with health data, a gateway for routing generated packets of Things to a remote server on the Internet based on tunneling, and a WebSocket server for analyzing health data in the Cloud. The experimental results show the efficiency of the proposed architecture regarding energy consumption, effectiveness and quality of service for IoT-based healthcare applications. However, it needs further improvements for data filtering and compression in the IoT part to gain better energy efficiency.

The authors of (Rault *et al.*, 2014) proposed an energy-efficient scheme for monitoring patients' health. The scheme focuses on reducing energy consumption at both the sensors and the users' mobile phones by using ZigBee and Bluetooth communications. Regularly, the sensors, select the suitable communication interface (i.e., either ZigBee or Bluetooth). If ZigBee interface is selected, then Bluetooth interface is switched off on the mobile phone side, which saves its energy. Meanwhile, the mobile phone runs Bluetooth scan regularly to detect sensors that need to communicate with it when there is no ZigBee communication. The simulation results show that the scheme improves energy efficiency without negatively affecting the performance. However, the scheme did not consider the communication with the Cloud, which also needs more solutions to achieve better energy efficiency and performance.

In (Papageorgiou, Zahn and Kovacs, 2014), a novel framework based on auto-configuration for Cloud-based IoT platforms is proposed. In fact, the framework benefits from the integration of system parameters, power consumption, in addition to auto-configuration algorithms. The results show the efficiency of the framework in terms of performance and energy consumption. However, the framework focuses only on using auto-configurations and other issues that affect energy consumption such as data transmission are not considered.

The authors of (Li *et al.*, 2017) proposed three-tier system architecture that composed of Things Tier, Fog Tier, and Cloud Tier. The Fog Tier is responsible for bringing processing and storage near to the IoT edge which reduces the communication overhead. Generally, the architecture aims to decrease the transmission latency, and bandwidth overhead by using Fog as a middleware between the Cloud and IoT. The results show the positive effect of Fog for gaining an efficient CoT system architecture regarding performance, communication latency, and energy consumption.

In (Dubey *et al.*, 2015), a service-oriented TeleHealth architecture based on Fog Data is presented. The use of Fog Data simplifies the process of gathering, storing and analyzing patients' vital data. The Fog computer is responsible for analyzing and filtering – i.e., trimming unnecessary raw data – the gathered data before transmitting them to the Cloud which significantly decreases the volume of the data that has to be stored and deployed to the Cloud, and at the same time improve the energy efficiency. The experimental results show that the architecture has made substantial improvements in energy consumption by reducing the communication and the amount of data between the Fog and Cloud as well as increasing the efficiency of the overall healthcare system.

Rani et al. (Rani *et al.*, 2015) proposed a novel scheme for an energy efficient IoT-based on Wireless Sensor Networks. Generally, the scheme focuses on reducing the energy consumption by leveraging the clustering concept, in which the whole network is subdivided into clusters of equal size. Direct communication with the upper cluster layers and between cluster nodes is not allowed except through cluster heads (CH) and cluster coordinators (CCO). The scheme also uses two algorithms: one for conserving energy by orchestrating the communication between the CCOs, and the other for optimizing the energy parameters. In other words, the scheme focuses on reducing the communication distance amongst the IoT objects to conserve energy, so the network lifetime is increased. The simulation results show that the scheme is a scalable and more energy efficient than the more popular schemes such as LEACH, SEP, and MODLEACH. However, the scheme requires further improvements in data transmission techniques and end-to-end delay so as to achieve greater efficiency.
The authors of (Huang *et al.*, 2014) introduced an energy-efficient deployment scheme for placing wireless sensor nodes (i.e., Things) in IoT. In this scheme, an algorithm for energy optimization is proposed based on the clustering concept and Steiner tree algorithm. The scheme increases the network lifetime by moving

the burden of direct communication to the relay nodes instead of sensing nodes. The results of the experiment reveal that the scheme is preferable to achieve green IoT deployment regarding network lifetime and energy consumption. However, the scheme ignores the application of compressed sensing approaches in gaining more energy efficiency, so further improvements are required.

The authors of (Abedin *et al.*, 2015) presented a system model for energy-efficient communications among IoT objects. The proposed system model employs the Cloud for IoT services deployment. Furthermore, the system model includes an energy efficient algorithm based on scheduling the activities of IoT sensors by using three major phases. The on-duty phase, in which the sensor works with its full capability, the pre-off duty phase to switch between on-duty and off-duty phases, and the off-duty phase, which is responsible for conserving energy based on three states: hibernate, sleep, and power off. The experimental results in a real IoT test-bed environment showed the efficiency of the system model regarding energy consumption and performance. However, the system model does not consider data transmission between sensors and the Cloud. Thus, further improvements are required, such as integration with Fog for more energy efficiency.

The authors in (Mao *et al.*, 2005) proposed a novel cluster-based approach known as EECS for saving energy and extends the wireless network lifetime. The EECS scheme increases energy efficiency during data gathering applications at regular intervals of time by selecting nodes with more remaining energy as cluster heads in a way that ensures optimal distribution of cluster heads. Furthermore, the EECS has a novel method for load balancing between the cluster heads. The simulation results show that EECS scheme prolongs the network lifetime by more than 35% when compared with LEACH.

In (Mehmood *et al.*, 2015), A. Mehmood et al. introduced a cluster-based mechanism, called EEMDC (energy-efficient multi-level and distance-aware clustering), for conserving energy in WSN. EEMDC divides the WSN region into three layers in a hierarchical structure according to the count-hop based distance from the base station. In EEMDC, idle listening is required due to the implementation of the levels in the clustering, thus, the cluster nodes turn to the active mode only when the control arrives at its level. The simulation results show that EEMDC is energy-efficient and it consumes less energy than other common cluster-based communication schemes (e. g. LEACH, MTE, and DDAR).

In (Praveena and Prabha, 2014), an approach for increasing energy efficiency based on level-k clustering hierarchy and single-hop communication (i.e., link-correlation) among nodes within the cluster is proposed. In this approach, the level-k cluster represents the top level while {k-1, k-2, … ,-2} represent the clusters of the underneath levels. Moreover, each node communicates with the corresponding cluster head that lies on the upper level in the hierarchy by using single-hop communication, while the cluster heads in level-k communicate through multi-hop communication. The proposed approach solves the problem of the bottleneck zone, the nodes that surround the sink node, thus they consume the energy due to their heavy traffic by letting these nodes create the level-k cluster head. The level-k cluster head represents the TDMA (Time Division Multiple Access) time slots for the corresponding level k-1 and likewise for the following levels of the cluster hierarchy. The TDMA technique helps create an organized cluster-based architecture which enhances the energy efficiency. The simulation results show the efficiency of this approach compared with LEACH, SEP, and DEEC in terms of stability, message delivery, and network lifetime. However, this approach works properly only in medium-sized WSN, so further improvements are required to be suitable for large-sized WSN as well as when integrating IoT-based WSN with the Cloud.

The authors of (Tang *et al.*, 2014) introduced a novel scheme for efficient data collection and aggregation called ECH-tree (Energy-efficient Hierarchical Clustering index tree). In general, ECH-tree splits the domain of the sensor network into even cell grids. Subsequently, the cell grids are clustered into sub-domains, which minimize the message broadcasting distance amongst them, and as a consequence reduce the energy consumption. Furthermore, the ECH-tree scheme contains a time-correlated querying method which answers end-user queries in a reduced power consumption manner. Moreover, ECH-tree eliminates data redundancy by changing the sensor data based on frequent time intervals, which allow the sink node to collect the query answers from the grid cell tables immediately. The experimental results confirm the energy efficiency of the ECH-tree scheme.

In (Yaacoub, Kadri and Abu-Dayya, 2012), a cooperative multi-hop approach for saving energy during data transmission is proposed. This approach organizes the whole sensor network into clusters that communicate with each other through multi-hop short range links, and the last cluster only communicates with the base station (BS) through a long range wireless link for conveying the aggregated multi-hop data. The simulation results show that significant energy efficiency could be attained by using an efficient multi-hop cooperation approach.

In (Etelapera, Vecchio and Giaffreda, 2014), a new approach is proposed for enhancing IoT energy efficiency based on re-configuring virtual objects (VOs) at runtime. In this approach, VOs are used as an abstraction to describe the semantics of ICT objects and the

related physical objects. In particular, the approach focuses on reducing energy consumption at transmission time by comparing the three types of compression modes, i.e., uncompressed, lossy, and lossless. The experimental results on

sample data captured from wireless weather stations show the ability of this approach to reducing the total energy consumption by up to 47.9% when changing the compression mode from uncompressed to lossy at runtime. However, this approach increases the latency of the provisioned service which affects the performance negatively.

A new method that improves the network lifetime using mobile sink nodes combined with the traveling salesman problem (TSP) algorithm is proposed in (Yu, Kim and Lee, 2013). In this method, the hotspot problem is solved, which consumed the energy of the sensor nodes that surrounded the sink, especially when using multi-hop communication because they act as a sensor and relay nodes at the same time, so the whole network collapses. By using a mobile sink with one hop communication, the transmission times can be reduced due to the wide range of coverage by each node. The simulation results show the efficiency of this method in improving the network lifetime and the energy efficiency of relay nodes.

In (Wankhade and Choudhari, 2016), an energy-efficient modified election-based Protocol (MEP) is proposed to increase the network lifetime. With MEP, sink nodes have the ability to decide which sensor nodes can represent the cluster heads (CHs) according to their additional energy, the remaining energy, and the node location. Besides, the CHs communicate with sink using the shortest path based on a congested link. Moreover, MEP is concerned by addressing the trade-off between energy efficiency and the QoS requirements. The simulation results show that MEP ensures energy-efficient routing in WSN as well as extending the network lifetime. However, further improvements are still required to gain energy efficiency in the whole IoT network regarding data transmission and processing.

Although there are different viewpoints about the energy consumption of Fog computing, the energy consumption of Cloud data centers can be reduced by using nano data centers (nDC) in Fog, especially in IoT applications (Jalali, Hinton, *et al.*, 2016). Based on this, the authors of (Jalali, Hinton, *et al.*, 2016) proposed two models, a flow-based and time-based model for shared and unshared network resources, respectively, after identifying the more exact scenarios for achieving better energy efficiency when using nDC in Fog rather than centralized DC. The experimental results revealed that nDC can help in achieving energy efficiency, according to the arguments of the designed system such as the type of applications and the amount of data preloading. However, this solution only works properly in specific applications such as in video surveillance wherethe number of accesses to nDC is not enormous.

In (Barcelo *et al.*, 2016), the authors propose a mathematical network and service models for optimizing the distribution of CoT-based services. The simulation results show the efficiency of this model for providing efficient IoT smart services in terms of low latency, flexibility, reliability and reducing the overall energy consumption by more than 80% when compared with other relevant proposals.

In (Alduais *et al.*, 2016), an approach for reducing data transmission time and size in IoT-based WSN is introduced. The approach benefits from the fact that in WSN, data transmission degrades energy efficiency more than performing instructions. To overcome this situation, the authors proposed a method to control the transmit (ON/OFF) radio frequency (RF) according to the current and last state values. Furthermore, an algorithm based on the relative differences between (single or multiple) current and last gathered value of sensors is proposed for minimizing the number of data packets transmitted. The simulation results showed that the efficiency of this approach to improve the performance and reduce the energy consumption as well as extend the network lifetime.

The work (Kaur *et al.*, 2017) proposed a container-based layered architecture, namely CoESMS, for the management of services at the Fog level. The primary objective is to increase the energy efficiency of the Fog nano data centers (nDC) by efficiently scheduling tasks on containers as well as migrating containers when desirable. To achieve this objective, the authors used two concepts, containers and game theory. Containers are distributed among various VMs, and tasks are scheduled to the appropriate containers based on the cooperative game theory. In this scheme, service tasks are scheduled on containers at Fog if they are marked as real-time tasks and if there are an adequate number of containers for processing. Further, to minimize energy consumption, internal and external migration of containers is accomplished when the upper or lower thresholds are violated. Compared to a container-based scheme that does not use CoESMS, the results showed that this scheme significantly reduces the energy consumption of nDC, and it ensures an acceptable service level agreement (SLA) to users.

Service composition is a concept that deals with integrating a number of services to fulfill a user's or application's request when a single service is not adequate to accomplish the job. However, the exchange of large amounts of data between these services has a negative impact on energy efficiency. To mitigate this, the authors in (Baker *et al.*, 2017) proposed an energy-aware service composition algorithm, E2C2, to select the optimal plan which has the least energy consumption without violating the user service level agreement (SLA). The algorithm focuses on finding the plan with the fewer services composition in order to minimize data transmission in a multi-cloud environment. Compared with other algorithms, such as All Clouds, the experimental results showed the efficiency of E2C2 in terms of energy consumption and performance.

Scheduling, load balancing and data replication are well-known energy-saving techniques. The purpose of scheduling (Fang, Wang and Ge, 2010) is to allocate

application/service tasks to suitable computing resources (i.e., in the Cloud or Fog environments) for execution. Load balancing (Oueis, Strinati and Barbarossa, 2015) focuses on keeping workloads balanced among all hosts or virtual machines to ensure that hosts/virtual machines are neither overloaded nor underutilized. The data replication (Verma, Sagar and Yadav, Aran Kumar and Motwani, Deepak and Raw, RS and Singh, 2016) concept focuses on bringing a copy of original data near end users (requests), to increase the system performance and fault tolerance, as well as decrease the burden on the servers and use of network bandwidth. Furthermore, server and tasks consolidation (Singh *et al.*, 2016) are used to save energy by offloading tasks from an underutilized server to another, followed by powering the former off, and accumulating tasks into a group followed by assigning such a group to a minimal number of virtual machines, respectively.

This section reviews the most important proposals for energy saving in both Cloud and Fog environments based on such resource management techniques, as task scheduling and load balancing. In (V. B. Souza *et al.*, 2016), a service allocation strategy model for Fog-to-Cloud (F2C) architecture is introduced. In the proposed model, the total capacity of Fog is divided into slots, while services are decomposed into atomic services; afterwards, such atomic services are allocated to the available slots. Furthermore, each slot is connected to one of the underlying devices for executing service requests. The model focuses on the allocation of services to Fog nodes to optimize service delays, processing loads and power consumption.

The authors of (Alnowiser *et al.*, 2014) introduced a mechanism for energy saving in a Cloud environment. They combined the dynamic voltage and frequency scaling (DVFS) technology with the reuse of virtual machines (VMs), live migration, and an enhanced weighted round-robin scheduling algorithm. To minimize energy use, the authors proposed three different algorithms for scheduling tasks, allocating/reusing VMs, and adjusting the CPU frequency. The

simulation results of different scenarios showed the appropriateness of the proposed mechanism to reducing the consumed energy and resource utilization.

A dynamic load balancing and consolidation algorithm for reducing the energy consumption and resource utilization in a Cloud environment is presented in (Sahu, Pateriya and Gupta, 2013). The authors designed an algorithm based on dynamic thresholds combined with a "compare-and-balance" technique. Using this technique, the algorithm performs either load balancing among hosts, or VM migration from one host to another. The simulation results showed the proposed algorithm's efficiency in energy consumption and resource utilization.

A systematic framework for solving the tradeoff between energy consumption and delays in a Fog-Cloud system is proposed in (Deng *et al.*, 2016). It calculates the power consumption and delays separately at Fog devices, Cloud servers, and WAN. Then, the total is obtained by aggregating such sub-components. The simulation results showed that the energy consumption by Fog devices increased due to increasing workload, whereas the computation delay of Cloud servers rose due to an increase in WAN communications. Therefore, an optimal workload is required to attain efficient load balancing with respect to energy consumption and delays.

In (Xuan-Qui Pham and Eui-Nam Huh, 2016), an algorithm for scheduling tasks in a Fog-Cloud computing environment is introduced. This algorithm performs scheduling by determining the priorities of tasks and then specifying the proper node for execution of each task. The experimental results showed the algorithm to be appropriate for balancing the tradeoff between performance and cost. However, the algorithm did not consider the energy consumption.

An algorithm combining data replication with load balancing is presented for a Fog-Cloud environment in (Verma, Sagar and Yadav, Aran Kumar and Motwani, Deepak and Raw, RS and Singh, 2016). According to this algorithm, user requests are processed at Fog-tier or Cloud-tier according to data availability. When the data are not available at Fog-tier, the request is forwarded

to the nearest Cloud server at Cloud-tier, which responds to the request in addition to replicating the requested data at the active Fog server for future use. A comparison of the proposed algorithm's results with those of other Cloud techniques (e.g., round-robin) showed this approach to perform efficiently; however, it was expensive.

A combination of Fog computing and microgrids equipped with local battery storage was applied to weather forecasting for renewable energy estimation (Jalali, Vishwanath, *et al.*, 2016). The IoT gateway decides to schedule the execution of tasks either in the Fog or the Cloud, according to the remaining energy in the local battery and the weather forecast status. For instance, if an IoT application requires intensive processing, the IoT gateway checks if the local energy is adequate for executing the process locally by awaking a local Fog device or sends the process to the Cloud otherwise. Moreover, the IoT gateway can perform the intensive tasks of sensors, reducing the energy consumed by sensors. The experimental results showed the combined use of Fog and microgrids to be energy-efficient. However, the desired efficient and dynamic energy management strategy will take advantage of such technologies to select the lowest power consumption choice of executing IoT applications in the Fog or the Cloud.

The authors of (Wang *et al.*, 2016) proposed a scheme, named CachinMobile, for caching frequently requested data at mobile phones of end users to improve the energy efficiency of Fog nodes. The scheme exploited the concept of social networks and device-to-device (D2D) communication to reduce the energy consumption of data transmission due to the use of short-range communications. To achieve this based on Evolved Node B (eNodeB)-gathered information, the authors selected end users with high social centrality (i.e., meeting a threshold value) to be the edge nodes for caching data. Subsequently, they applied a genetic algorithm to determine the optimal data placement. The simulation

results showed that the scheme was more efficient in energy consumption and performance than other related non-caching and randomly caching schemes.

The work presented in (Wen *et al.*, 2017) proposed a Fog orchestration framework based on a parallel genetic algorithm (GA-Par). The framework focused on guaranteeing the best optimization for selection and placement of fog resources and IoT appliances to organize an application workflow, while ensuring quick response times and quality of service (QoS). The simulation results showed GA-Par to be more efficient than a standalone genetic algorithm (SGA). However, the framework faced a scalability challenge when dealing with an increased number of Fog devices and requests.

An algorithm for the optimal cluster creation and load balancing in Fog computing is proposed in (Oueis, Strinati and Barbarossa, 2015). First, the local computing resources are allocated to small cells, i.e., Fog resources, to serve mobile users' requests locally. Subsequently, small-cell Cloud clusters are created to process requests that require resources exceeding those available in small cells of the Fog. Moreover, the authors designed three types of the algorithm with metrics varying according to the design objective, such as latency or energy efficiency. The analytical results showed that the proposed algorithm was more effective than non-clustering and static clustering techniques, while ensuring the user's satisfaction (i.e., QoS).

Based on proposals available in the related literature, a new energy-aware allocation strategy for placing application modules (tasks) on fog devices is proposed in the next section.

All of the solutions mentioned above to improve the energy efficiency issue are summarized in Table 2.3.

Table 2.2: Summary of the most relevant proposals regarding energy efficiency

Reference	Method	Description	Benefits	Limitation
Petrolo et al. [63]	Gateway architecture	The gateway is based on the combination of virtualization technologies and prediction algorithms on data production	Energy efficient and enhanced CoT-based service deployment, management, and allocation	Further adoptions in prediction algorithms on data production to decrease the communication among the gateway and Things are needed
Chang et al. [22]	Distributed cluster-based scheme	The scheme utilizes the concept of distributed clusters to increase the network lifetime by reducing the data transmission distance among sensor nodes	Improving energy efficiency, network lifetime, and performance	Additional overhead in cluster head (CH) node selection. Did not consider communication with the Cloud.
Kaur et al. [64]	Energy-efficient architecture for IoT	Architecture focuses on reducing energy consumption on IoT by using efficient sleep mode	Energy-efficient IoT regarding sensing, transmission, analytics, and sharing info with apps.	Further adoptions are needed to address downstream traffic scheduling issue which degrades user service.
Mohsen Nia et al. [65]	Energy-efficient scheme for health monitoring	The scheme is based upon compressive sampling for efficient continuous personal health monitoring	Energy efficiency in data collection, transmission, and storage	The scheme did not consider the Cloud side, so further adoptions are needed
Mangali et al. [66]	Energy-efficient scheme for eHealth monitoring	Cluster-based scheme, in which WSN is divided into clusters, each cluster has a cluster head for routing and executing specific tasks.	Energy efficiency and increases the WSN lifetime	The scheme ignores the Cloud side which is also affects power consumption negatively, so further adoptions are needed
Granados et al. [67]	IoT Healthcare architecture	The architecture is based on smart gateways that use thePower over Ethernet (PoE) standard for effective data transmission and to supply IoT sensors with power	Energy-efficient solution for healthcare and improves the performance by moving the process from IoT sensors to the Cloud via the smart gateway	Efficient only in the case of wired sensors and fixed scenarios, so it needs further modification to be suitable for wireless sensors and unfixed scenarios
Gia et al. [68]	IoT eHealth architecture based on customized 6LoWPAN	The architecture benefits from the effective customization of 6LoWPAN for improving the energy efficiency and reliability of IoT-based eHealth applications	Efficient regarding energy consumption, effectiveness, and QoS	It needs further improvements regarding data filtering and compression in the IoT part to gain better energy efficiency
Rault et al. [69]	Health monitoring scheme based on ZigBee and Bluetooth	The scheme focuses on reducing energy consumption at both sensors and user mobile phones, by using ZigBee and Bluetooth communications	Improves energy efficiency without negatively affecting the performance	It ignores the communication with the Cloud which also needs more solutions to achieve better energy efficiency and performance
Papageorgiou et al. [70]	A novel framework based on efficient auto-	The framework benefits from the integration of system parameters, power consumption, in addition to auto-configuration	The framework is efficient regarding energy consumption and performance	It only focuses on exploiting efficient auto-configurations, and other issues that affect energy consumption such as

	configuration for Cloud-based IoT platforms	algorithms		data transmission were not consider
Li et al. [71]	Three-tier CoT system architecture	The architecture aimed to decrease the transmission latency and bandwidth overhead by using Fog as a middleware between the Cloud and IoT	Significant efficiency regarding performance, communication latency, and energy consumption.	It focuses only on reducing communication overhead and ignores the process of data collection in the IoT part
Dubey et al. [72]	Service-oriented TeleHealth architecture based on Fog Data	Uses Fog Data to simplify the process of gathering, storing and analysis of patients' vital data	Substantial improvements in energy consumption by reducing the communication and the amount of data between the Fog and Cloud. Increases the efficiency of the overall healthcare system	Further improvements in the IoT-tier are required
S. Rani et al. [73]	A novel cluster-based scheme for energy efficient IoT	Subdivides the IoT network into equal size clusters. Then, uses two algorithms to orchestrate CCO communications and optimize the energy efficiency parameters	Provides scalability and energy efficiency	Further improvements in data transmission and end-to-end delay to increase energy efficiency
J. Huang et al. [74]	Energy-efficient deployment scheme	Scheme for placing wireless sensor nodes in IoT based upon algorithm for energy optimization	Provides green IoT regarding network lifetime and energy efficiency	Did not consider applying compressed sensing approaches for more energy efficiency
S. F. Abedin et al. [75]	System model for energy efficient communications among IoT objects	Exploits the Cloud for IoT services deployment. Besides using algorithms to schedule IoT sensor activities to save energy	Improves energy efficiency and performance	Further improvements in data transmission between IoT sensors and the Cloud
M. Ye et al. [76]	A novel cluster-based scheme EECS	Selects CHs according to the remaining energy and ensures optimal distribution for CHs and load balancing amongst them	Prolongs the network lifetime	Did not consider the communication with the Cloud
A. Mehmood et al. [77]	Cluster-based Approach EEMDC	Conserves energy efficiency in WSN by dividing it into three hierarchical layers based on the count-hop	More energy efficient than cluster-based communication schemes	Did not consider the communication with the Cloud
N. G. Praveena and H. Prabha [78]	Level-k Clustering based energy efficiency scheme	Uses level-k clustering hierarchy besides single-hop communication among nodes in the cluster and multi-hop communication among CHs in level-k	Efficient regarding stability, message delivery, and network lifetime	Works properly in medium-sized WSN and requires integration with the Cloud
J. Tang et al. [79]	Novel scheme (ECH-tree) for data aggregation and collection	Minimizes the message broadcasting distance by splitting the SN into clustered sub-domains	Reduces energy consumption and eliminates data redundancy	Did not consider the scenario of integration with the Cloud

E. Yaacoub et al. [80]	Cooperative multi-hop approach	Organizes SN into clusters that communicate via multi-hop short-link range and the last cluster communicates with sink through a long range wireless link	Provides significant energy efficiency	Did not consider the scenario of integration with the Cloud
M. Eteläperä [81]	A new approach for enhancing IoT's energy efficiency	Approach based on re-configuring virtual objects at runtime and focuses on reducing energy at transmission time	Reduces energy consumption up to 47.9%	Increases latency which degrades the performance
J. Kim and J. Lee [82]	A new approach for increasing network lifetime	Uses mobile sink nodes combined with a TSP algorithm	Improves network lifetime and energy efficiency of relay nodes	Did not consider the scenario of integration with the Cloud
N. R. W. Prof and D. N. Choudhari [83]	An energy-efficient modified election-based Protocol (MEP)	MEP allows sink nodes to choose the CHs according to specific criteria. Also, sink nodes communicate with CHs using the shortest path	Energy efficient routing in WSN, increases network lifetime	Further improvements to gain energy efficiency in the whole IoT network regarding data transmission and processing
F. Jalali et al. [84]	A Fog-based approach for reducing energy consumption of Cloud data centers	Utilizes nano data centers in Fog by exploiting flow-based and time-based models	Achieves energy efficiency at centralized DCs according to some criteria of the designed system	Only works properly in specific applications or services
M. Barcelo et al. [41]	A mathematical network and service models	Mathematical models to solve service distribution problem based on linear programming	Efficient services regarding low-latency, reliability, flexibility, and reduces the overall energy consumption	Further improvements are required such as computational complexity issues
Alduais, N A M; et al. [85]	A method for reducing number of packet transmissions and their size in IoT-based WSN	An approach based on relative differences among current sensors values and last gathered values to switch RF transmit to ON/OFF	Improves energy efficiency and performance as well as network lifetime	Did not consider data transmission to the Cloud
K. Kaur et al. [86]	A container-based layered architecture (CoESMS)	An architecture to manage services at the Fog level by using efficient task scheduling based on game theory	Increases energy efficiency at Fog nano datacenters. Acceptable Service Level Agreement (SLA) to users	
T. Baker et al. [87]	An energy-aware service composition algorithm (E2C2)	The algorithm selects the optimal plan which has the least energy consumption without violating the user's service level agreement.	Efficient in terms of energy consumption and performance	

2.6 Discussion and Open Issues

Although CoT paradigm offers new opportunities for IoT-based smart services, several new challenges or open issues have arisen. The most important issues as explained in (Botta, De Donato, *et al.*, 2016; Cavalcante *et al.*, 2016a; Díaz, Martín and Rubio, 2016b) are energy efficiency, the lack of standardization (for architecture as well as data and services), efficient big data management and analytics, security and privacy, Fog computing, scalability, and mobility support. However, this research mainly focuses on the energy efficiency, so all of the remaining issues will be explained in brief.

2.6.1 Discussion

In this subsection, a comparison regarding the energy efficiency of the works mentioned in Section 2.6. Table 2.4 summarizes the comparison parameters: the techniques used, place of executing the scheme, renewable energy, traffic overhead, performance, scalability, and quality of service (QoS) for each analyzed proposal. As seen in Table 2.4, the techniques used range from clustering concept to Fog-based approaches. The recent works, such as (Kaur *et al.*, 2017) tried to tackle the service migration in an energy efficient manner at the Fog level. However, this trend is still immature and needs further research. Furthermore, the comparison of the results shows that some of the proposed solutions concerned the WSN part while the others tried to reduce the energy consumption only in Cloud datacenters. Therefore, energy-efficient schemes that increase the energy efficiency of the whole system are required. Moreover, the comparison of the results also reveals that most of the proposed approaches

focused on reducing energy consumption and ignored the other important metrics, such as scalability.

In summary, all of the aforementioned schemes show the importance of energy efficiency in gaining efficient CoT-enabled services and applications regarding availability, reliability, and performance. Moreover, most of the proposals tried to tackle the energy efficiency issue only in the IoT part and ignored the communication with Cloud scenarios. Therefore, the energy efficiency issue needs further efforts, particularly in delay-sensitive services such as smart Healthcare.

Table 03: A comparison among the available energy efficiency proposals

Reference	**Used techniques**	**Scheme place**	**Renewable energy**	**Traffic overhead**	**Performance**	**Scalability**	**QoS**
Petrolo et al. [63]	Virtualization technologies, and prediction algorithms	Gateway	No	No	Acceptable	Yes	Yes
Chang et al. [22]	Distributed clustering, and Data compression	WSN	No	No	Acceptable with small overhead	Yes	Yes
Kaur et al. [64]	Sleep mode	IoT sensors	No	No	Acceptable	Yes	Low, so further adoptions are needed.
Mohsen Nia et al. [65]	Compressive sampling	WBAN	No	No	Acceptable	Yes	Low
Mangali et al. [66]	Clustering	WSN	No	No	Acceptable	Yes	Low
Granados et al. [67]	Power over Ethernet	Gateway	No	No	Acceptable	No	Yes
Gia et al. [68]	Customized 6LoWPAN	WAN	No	No	Acceptable	Yes	Yes
Rault et al. [69]	ZigBee and Bluetooth	WAN	No	No	Acceptable	No	Yes
Papageorgiou et al. [70]	Auto-configuration algorithms	IoT platform	No	No	Acceptable	Yes	Yes
Li et al. [71]	Fog computing as a middleware tier	Whole system architecture	No	No	Very good	Yes	Yes
Dubey et al. [72]	Fog Data	Whole system architecture	No	No	Very good	Yes	Yes

S. Rani et al. [73]	Clustering and orchestration algorithms	IoT network	No	No	Acceptable	Yes	Yes
J. Huang et al. [74]	Clustering and Steiner tree algorithm	WSN	No	No	Acceptable	Yes	Yes
S. F. Abedin et al. [75]	System model with scheduling algorithms	Whole system model	No	No	Acceptable	Yes	Yes
M. Ye et al. [76]	Clustering and load balancing algorithm	Wireless network	No	No	Acceptable	Yes	Yes
A. Mehmood et al. [77]	Clustering	WSN	No	No	Acceptable	Yes	Yes
N. G. Praveena and H. Prabha [78]	Level-k Clustering and TDMA (Time Division Multiple Access) technique	WSN	No	No	Acceptable	No	Yes
J. Tang et al. [79]	Clustering index tree and time-correlated querying technique	WSN	No	No	Acceptable	Yes	Yes
E. Yaacoub et al. [80]	Cooperative multi-hop approach and clustering	WSN	No	No	Acceptable	No	Yes
M. Eteläperä [81]	Re-configuring virtual objects and compression technique	IoT network	No	No	Acceptable	No	Low
J. Kim and J. Lee [82]	Mobile sink nodes combined with traveling salesman problem (TSP) algorithm	WSN	No	No	Acceptable	Yes	Yes
N. R. W. Prof and D. N. Choudhari [83]	Election-based method for choosing the optimal sink node to be the CH node	WSN	No	No	Acceptable	Yes	Yes
F. Jalali et al. [84]	Fog computing combined with flow-based and time-based models	Fog nano datacenters	No	No	Very good	No	Yes
M. Barcelo et al. [41]	Mathematical network and service models	Network	No	No	Acceptable	Yes	Yes
Alduais, N A M; et al. [85]	A method to control radio frequency transmission and algorithm to reduce the transmitted data packets	WSN	No	No	Acceptable	Yes	Yes
K. Kaur	Containers migration	Fog level	No	No	Very good	Yes	Yes

et al. [86]	and game theory based task scheduling						
T. Baker et al. [87]	Service composition algorithm (E2C2)	Multi-cloud environment	No	No	Acceptable	Yes	Yes

2.6.2 Open Issues

Lack of Standardization

The absence of standardization is considered as one of the most critical issues that face CoT. This issue is due to the lack of standards in both IoT and Cloud computing models (Fortino *et al.*, 2018). Indeed, the currently used web-based APIs (e.g. RESTful APIs) only facilitate the communication between the Things and Cloud, however, they are not designed to deal with M2M (machine-to-machine) interactions (Botta, De Donato, *et al.*, 2016; Cavalcante *et al.*, 2016b). Furthermore, the literature shows that the majority of relevant CoT-based proposals do not adhere to a unified standard in the process of development due to the lack of standardized architecture. In this context, reference architecture (RA) (Botta, De Donato, *et al.*, 2016), which defines the standards and rules that developers should follow for building CoT-based solutions, can help alleviate the complexity of development. Moreover, RA can help the deployment of efficient CoT standards that are capable of supporting smart services with optimized M2M communications (Cavalcante *et al.*, 2016b).

Security and Privacy

This issue is inherited from security and privacy in the Cloud and IoT, besides the newly arisen vulnerabilities from the integration between them. From the Cloud point of view, to obtain transparent access to stored data in Cloud requires a third-party, which represents a threat that may exploit the data for malicious purposes. Meanwhile, IoT devices implement simple security techniques due to

their constrained resources, thus, they represent an easy target for intruders. These security breaches ranging from unauthorized changes of sensor data to service denial (DoS attack) [15], [33], [34], [37]. Specifically, CoT-based healthcare requires a strong protection for patient health data, which are highly sensitive. Therefore, it is important to protect health information in different layers, i.e., IoT devices layer, Network layer, and Cloud layer. In this context, using distributed security architecture with lightweight cryptography schemes may help to achieve protected health information in the IoT environment as well as Cloud (Ben Ida, Jemai and Loukil, 2016). Finally, security and privacy represent a continuous challenge that needs efficient solutions, in which the consumers of CoT-based services can trust, so keeping the flourish of such services.

Big Data Management and Analytics

Despite the ability of the Cloud to manage big data due to its virtually unlimited storage and processing capabilities, dealing with IoT generated big data still represent a challenge regarding the Cloud shortcomings in support of IoT devices. Most of the proposed solutions to tackle this issue reveal their inappropriateness regarding data handling and performance efficiency [15], [59]. Furthermore, handling a massive amount of data generated by IoT sensors in the Cloud is inconvenient regarding the constrained bandwidth, intolerable delay, and security. Therefore, efficient solutions based on innovative paradigms, e.g. Fog computing, that work in collaboration with the Cloud are necessary (Zhang *et al.*, 2017).

Fog Computing

Although the Cloud of Things paradigm successfully solves the majority of the IoT-related issues totally or partially, there are still issues that represent a

challenge, such as mobility support and low latency [21]. From this point of view, Fog Computing can effectively help to meet such requirements (Díaz, Martín and Rubio, 2016b; Masip-Bruin *et al.*, 2016). In fact, Fog Computing is an innovative and distributed computing model, through which all the smart services are executed at the network edge. In Fog Computing, edge devices (Fog nodes) are equipped with more storage and processing capabilities, besides functionalities such as edge analytics (Masip-Bruin *et al.*, 2016). However, there are several issues with the IoT, Fog, and Cloud layered architecture (i.e., three-tier CoT architecture) regarding optimized scheduling, fog networking management, security and privacy (Stojmenovic, 2015; Yi, Li and Li, 2015).

Quality of Service Assurance

Ensuring an acceptable level of QoS for CoT-based applications is still challenging due to the combination of different and heterogeneous technologies. To achieve QoS, it is crucial to precisely identify the QoS metrics at each CoT layer. Furthermore, CoT-based real-time applications/services require end-to-end QoS. Therefore, QoS-aware CoT architecture represents an open issue that needs efficient solutions (Aazam *et al.*, 2014; Botta, De Donato, *et al.*, 2016; Shah *et al.*, 2016).

Unpredictable Latency

Unpredictable latency is one of the most significant challenges, that is, intolerable for delay-sensitive services, e.g. healthcare, that require timely response (Nan *et al.*, 2016). In fact, this latency appears due to the massive communications with the Cloud for data storage or processing purposes. Moreover, the latency also has a negative impact on QoS. To address this issue, the authors of [31], [47] proposed to exploit the strategic position of smart gateways by integrating them with Fog paradigm, in order to provide end users with services at the network edge. Finally, Fog proved its suitability in solving latency, however, additional

efforts are required to carry out a performance evaluation of the whole Fog-enabled smart gateway CoT system model.

Energy Efficiency

Energy efficiency is a great concern, especially in smart things and data centers, due to its negative effect on quality of service (QoS), operating costs, and the environment (Dabbagh *et al.*, 2015; Hassanalieragh *et al.*, 2015). For such reasons, effective solutions that minimize energy consumption in both data centers and Things as much as possible are required. Furthermore, reducing the communication (i.e., trimming unnecessary communication) and data transmission between Things and Cloud datacenters can lower energy consumption

significantly (Chang, 2014; Dabbagh *et al.*, 2015). Effective scheduling techniques can also be utilized to achieve energy efficient IoT communications (Prasad and Kumar, 2013). Some solutions mentioned that effective sleep mode implementation, besides using a smart gateway that utilizes Fog computing to bring Cloud resources near to IoT objects, can enhance energy efficiency (Koubaa and Shakshuki, 2015). In this context, Fog is used to store Things data, whereas the gateway decides either to submit data to the Cloud or not according to the acknowledgment of the consumer application or service (Aazam and Huh, 2014; Koubaa and Shakshuki, 2015). In addition, heavyweight security schemes for authentication, Key establishment and distribution have a significant impact on energy consumption of IoT resource-constrained devices, so lightweight security schemes are required (Saied *et al.*, 2014). Furthermore, game theory can be used as a mechanism for analyzing WSN, in the case of the interaction between wireless sensor nodes with the competitive nature for acquiring the constrained network resources. With various types of models (i.e., non-cooperative games and cooperative-enforcement games), game theory can efficiently reduce the consumed energy of data aggregation processes without

affecting the network lifetime negatively (Alskaif, Guerrero Zapata and Bellalta, 2015). However, energy efficiency is still an open challenge that needs further research.

2.7 Summary

Currently, Cloud and IoT are extensively applied in several information technology applications such as healthcare and smart cities. However, reliable CoT-based services – particularly, highly delay-sensitive services such as Healthcare – require energy-efficient CoT architectures. This chapter surveyed CoT architectures, platforms, and their implementation in Healthcare. Furthermore, the chapter explained the CoT related issues, in brief, since it mainly investigated the energy efficiency issues with the more relevant proposals in detail. This investigation showed that the majority of the proposals were not concerned about energy efficiency when dealing with IoT Cloud scenarios. Therefore, efficient solutions for obtaining energy efficiency in both data processing and transmission are still required. Moreover, the new solutions should balance the trade-offs amongst energy efficiency, quality of service (QoS), and performance.

CHAPTER III

METHODOLOGY

3.1 Introduction

The convergence of the Cloud and IoT (CoT) represents a vision of the future Internet and supports a new and richer portfolio of smart services. However, such combination does not fit services with real-time, delay-sensitive and energy efficiency requirements due to the inherent limitations of these aspects of the Cloud. To this end, Fog computing was proposed by Cisco for providing such services at the network edge (Cisco Systems, 2016). Many benefits can be attained by using Fog computing, such as reducing the energy consumption and the load on data centers, in addition to conserving the network bandwidth (V. B. Souza *et al.*, 2016; Li *et al.*, 2017).

A recently proposed integration of Fog computing with IoT-based healthcare, see Figure3.1, represents a new trend in innovative e-health solutions. This combination enables healthcare services with improved latency, energy consumption, mobility, and Quality of Service (QoS). Such improvements result from the key characteristics of Fog, such as (*i*) proximity to end-users/IoT devices, and (*ii*) Mobility support of end-users, enabled by a geographically distributed architecture. The proximity to end users supports real-time responses and reduces latency, whereas mobility significantly promotes ubiquitous healthcare by enabling patients to obtain healthcare services efficiently regardless of location. As a Fog server can process the data gathered from IoT devices without reliance on the Cloud, it can effectively save the network bandwidth and cloud storage for vital data and processes (Shi, Ding, Wang, Roman, *et al.*, 2015; Andriopoulou, Dagiuklas and Orphanoudakis, 2017).

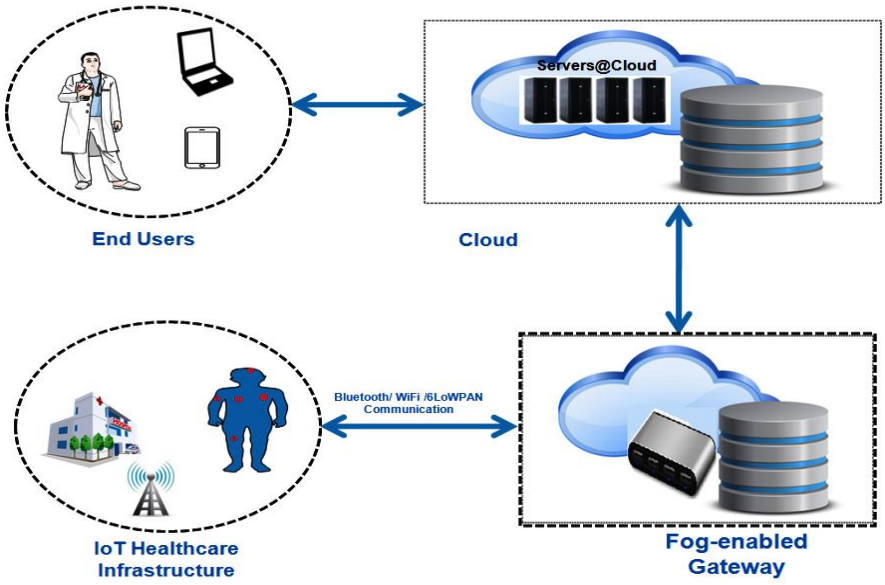

Figure 3.1.An illustration of Fog-integrated CoT-based healthcare architecture.

- To achieve an efficient scheduling and resource allocation in a Fog-enabled CoT system, suitable metrics should be considered according to the required objectives, such as application types, user mobility, and energy efficiency. Accordingly, the scheduling strategy should determine the location where applications or tasks are offloaded, either the Fog or the Cloud. It also determines the priority of applications' execution at a particular Fog instance according to the respective delay constraints. Furthermore, the scheduling strategy should consider various possible scenarios of application tasks execution, taking the primary objectives into account (Bittencourt *et al.*, 2017). The primary contribution of this chapteris to propose a Fog-enabled CoT system model along with an allocation strategy to reduce the energy consumption of Fog devices (i.e., Fog servers) based on the remaining CPU capacity and available stored energy, while ensuring efficient performance of real-time task execution.

Based on proposals available in the related literature, a new energy-aware allocation strategy for placing application modules (tasks) on fog devices is proposed in the next section.

3.2 The Proposed Energy-Aware Model

The approaches mentioned in the literatureshowed the significance of energy efficiency in obtaining reliable IoT services. This section briefly describes the system model and proposes an energy-aware allocation policy for placing application modules in Fog devices (Fog computing devices). The policy selects energy-efficient Fog devices more frequently than energy-inefficient ones.

3.2.1 System Model

As mentioned in (Li *et al.*, 2017), the Fog-enabled CoT system model is composed of Things-tier, Fog-tier and Cloud-tier. The Things-tier comprises heterogeneous physical entities organized into clusters according to geographic regions. The physical objects are provisioned as services through virtual entities allocated in Fog and Cloud tiers. The Fog-tier includes geographically distributed Fog instances (FIs) located alongside the Cloud and the continuum of things as depicted in Fig. 3.2. Such FIs communicate with IoT-tier and Cloud-tier through IoT gateways and Fog (Fog-to-Cloud) gateways, respectively. Finally, the Cloud-tier is composed of heavy-duty datacenters for processing and persistent storage.

The application model is illustrated by a directed acyclic graph (DAG), where vertices (V) denote the tasks or application modules, while edges (E) denote the dependencies among application tasks. Each application task has a type (e.g., sensing, actuating and processing) and workload. The workload identifies the resources required to execute the task, such as the CPU capacity. The optimal place for application task deployment should be specified accordingly, i.e., the Fog or the Cloud.

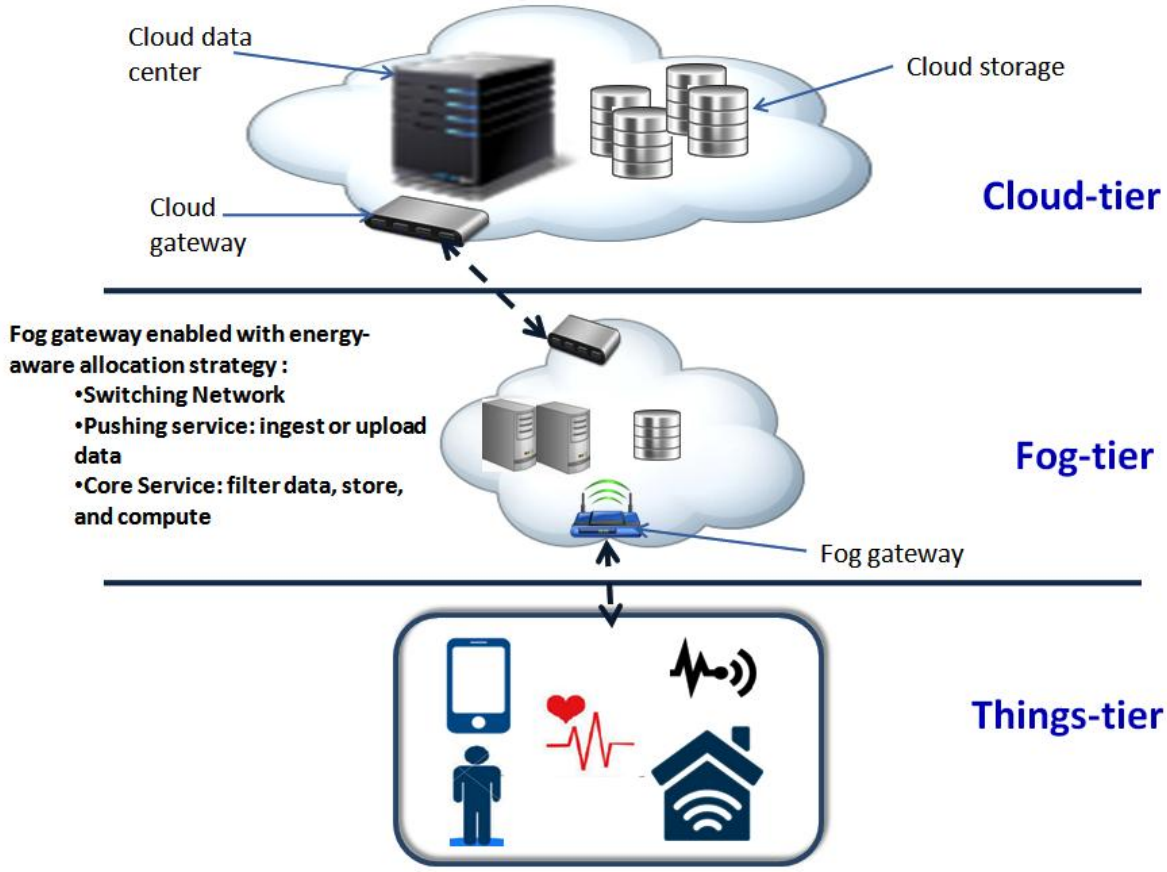

Figure 3.2. The three-tier CoT system model, in which data and tasks are handled in the appropriate place either Fog or Cloud.

3.2.2 Proposed Algorithm

The proposed approach (shown in Algorithm 1) is an energy-efficient strategy that allocates the incoming application modules (tasks) to Fog devices based on the remaining CPU capacity and energy consumption. The default placement strategy, called edge-wards, considers CPUs available for placement. For instance, if the current Fog device in the path does not meet the application module processing demands, the edge-wards strategy forwards it upwards until it finds a suitable Fog device or reaches the root, i.e., the Cloud. The main objective of the proposed allocation strategy is to increase the energy efficiency at Fog devices by allocating application tasks based on improved round robin (IRR) and dynamic voltage and frequency scaling (DVFS) algorithms. In other words, it places the application module on a Fog device that ensures a minimum

increment of energy consumption. The policy selects energy-efficient fog devices more frequently than energy-inefficient ones. The DVFS technology is used to adjust the CPU frequency of Fog computing devices (Fog servers) in a way that guarantees a reduction in energy consumption. Furthermore, the strategy tries to use Fog devices efficiently. In other words, it allocates the workload among Fog devices in a balanced way that ensures that a Fog device is neither underused nor overloaded. To this end, the strategy works together with the edge-ward placement strategy to guarantee that delay-sensitive tasks are placed in Fog devices as much as possible.

Algorithm 1: Energy-aware allocation of application modules

Input: *fogDevicesList, modulesToPlaceList*

Output: *placedModulesList in an energy efficient manner*

1. Fog devices (Fog servers) implement *DVFS* to adjust their CPUs frequencies
2. allocatedDevice = NULL
3. **for each** *fogDevice* in fogDevicesList **do**
4. **for each** *module* in modulesToPlaceList **do**
5. estimateConsumedEnergyAfterAllocation (*fogDevice*, *module*)
6. **if** *fogDevice* is suitable for *module* **then**
7. allocatedDevice = *fogDevice*
8. **else**
9. search for *fogDevice* upwards
10. **end**
11. place *module* on allocatedDevice
12. **end**
13. **end**

Algorithm 2: Adjusting CPU frequency using DVFS

Input: Tasks = { t_1 , t_2 , t_3 ,....., t_n}

Number of instructions = {$(NOI)_1$, $(NOI)_2$, $(NOI)_3$,....,$(NOI)_n$}

Deadline for each task { d_1 , d_2 , d_3 , ... , d_n}

FrequencySet{ } = the discrete values of the CPU frequency.

Output: Adjust the CPU frequency (cpuF) to the optimal frequency

1. **For** i = 0 to N **do**

2.Calculate the Optimal Frequency $(OF)_i$ for t_i

3. **(**$OF)_i = (NOI)_i / d_i$

4. **if** $(OF)_i$ ϵ FrequencySet{ }**then**

5. cpuF = $(OF)_i$

6. else

7.**for** k = 0 to M **do**

8. **If** ($(OF)_i$< $(FrequencySet)_k$)**then**

9. cpuF = $(FrequencySet)_k$

10.**endfor**

11. **endfor**

Algorithm 3: Estimate the ConsumedEnergyAfterAllocation

Input: *fogDevicesList, modulesToPlaceList*

Output: *the most suitable fogDevice for module allocation*

1. **for each** *fogDevice* in fogDevicesList **do**
2. **for each** *module* in modulesToPlaceList **do**
3. **calculate** the potential utilization MIPS when allocating the *module* on the *fogDevice*by adding the previous utilization MIPS to the current utilization MIPS
4. **calculate** the potential energy consumed on *fogDevice*based onthe potential utilization MIPS
5. **return** the potential energy consumed by*fogDevice*
6. **end**
7. **end**

3.3 Summary

This chapter studied the importance of the interplay between Fog computing and the two-tier CoT paradigm for achieving reduced latency and energy consumption. It also proposed an energy-aware allocation policy for placement of application modules (tasks).

CHAPTER IV

SIMULATION TOOL AND THE IMPLEMENTATION

4.1 Introduction

This chapter briefly describes the simulation tool used to evaluate the performance of the proposed approach. It also describes the case study scenario used to perform the study and depicts it as a directed acyclic graph.

4.2 Simulation Tool

Fog-enabled CoT scenarios can be simulated using available tools, e.g., DEVS (discrete event system specification) (Etemad, Aazam and St-Hilaire, 2017), SimPy (Li *et al.*, 2017), and iFogSim (Gupta *et al.*, 2017). As stated in [6, 25], iFogSim is the most suitable tool for simulating application environments that combine IoT, Fog, and Cloud. Such suitability arises from the following features of iFogSim:

- It is implemented on top of CloudSim, a popular tool for simulating Cloud environments, extending its most relevant components, such as the datacenter and Cloudlets.
- It is the first simulator of IoT objects, such as sensors, connecting them to Fog nodes and the Cloud in a hierarchical architecture.
- It is suitable for studying and evaluating such various aspects of Fog-enabled CoT applications, as latency, mobility and energy efficiency.

4.3 Case Study Scenario

Remote patient monitoring (RPM) is used as a case study to evaluate the energy efficiency of the proposed placement strategy. In fact, RPM shifts the traditional hospital-centric approach to monitoring patients with chronic diseases to a more

efficient patient-centric one. As a consequence, patients can independently obtain care regardless of location, while their families and caregivers are engaged remotely. In general, an RPM system comprises three primary modules: data acquisition, diagnostics (concentrator), and visualization. To acquire data, the patient is equipped with wearable or implantable body sensors, i.e., using body area networks (BANs). After being acquired by a sensor, the collected data are sent through the patient's smart phone to the diagnostic module for processing. Finally, the calculated results or analytics are displayed in an understandable form by the visualization module. In this case study, the focus is on RPM for monitoring patients with diabetes. In this scenario, patients' physiological parameters, such as blood pressure (BP), blood glucose (BG) or Glycaemia, and weight scale (WS) should be collected by the appropriate sensors. The application model of this scenario is depicted as a directed acyclic graph, as shown in Figure 4.1. For simplicity, the patient is only equipped with the blood glucose sensor. In the RPM use case, The configurations of the inter-module edges, fog devices, sensors are described in Table 4.1, Table 4.2, Table 4.3, respectively.

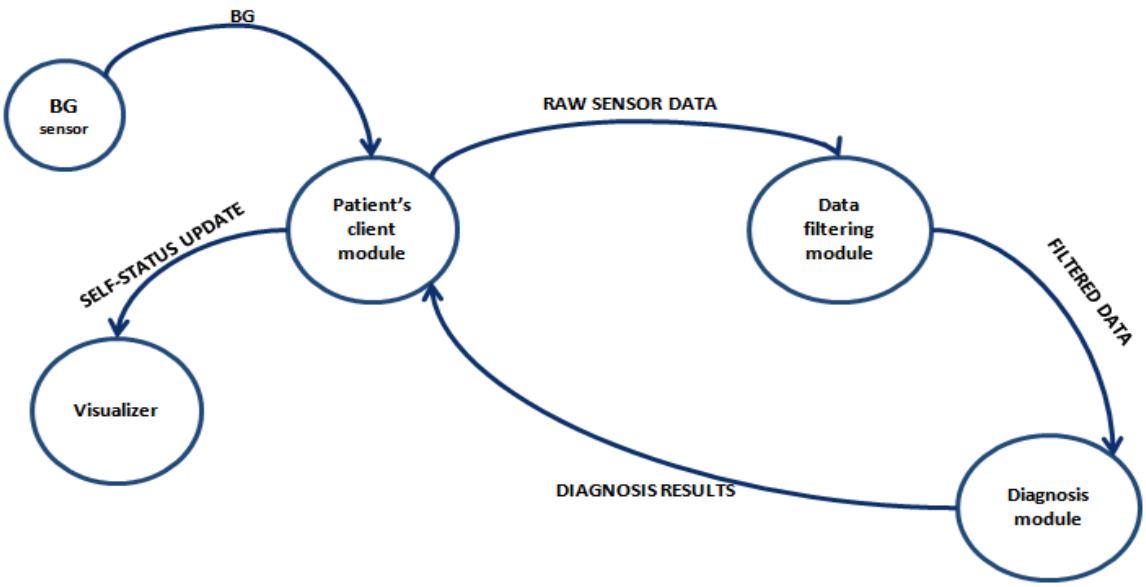

Figure 4.1. The application model of the RPM scenario depicted as a directed acyclic graph.

Table 4.1: Description of inter-module edges in the RPM application

Tuple type	**CPU length (MIPS)**	**N/W length**
BG	2000	500
_SENSOR	3500	500
DIAGNOSIS_RESULT	14	500
SELF_ STATE_ UPDATE	1000	500

Table 4.2: Configuration of fog devices for RPM application

Device type	**CPU (GHz)**	**RAM (GB)**	**Energy consumption (W)**
Cloud VM	3.0	4	107.339(M) 83.433(I)
Wi-Fi gateway	3.0	4	107.339(M) 83.433(I)
Smartphone	1.6	1	87.53(M) 82.44(I)
ISP gateway	3.0	4	107.339(M) 83.433(I)

M = the energy consumed when modules placed on devices

I = the energy consumed in devices without modules placement.

Table 4.3: Configuration of sensors for RPM application

Sensor	**Tuple CPU length**	**Average inter-arrival time (ms)**
Blood glucose (BG)	2000 million instructions	10

4.4 Summary

In this chapter, energy-aware Fog-enabled Cloud of Things modelalong with allocation algorithm for placement of application modules on Fog devices have been proposedfor healthcare. Furthermore, the simulation tool and the use case have been also identified.

CHAPTER V

RESULTS ANALYSIS AND EVALUATION

5.1 Introduction

This chapter analyses the results obtained in the above case study simulation scenario, with and without the proposed allocation policy and with and without Fog (i.e., using a two-tier CoT). The analysis recognizes the efficiency aspects of energy consumption, latency and network bandwidth usage. To obtain more accurate results, the following four physical topology configurations with different workloads are considered: *i*) 2 Fog devices and 4 smartphones; *ii*) 4 Fog devices and 4 smartphones; *iii*) 4 Fog devices and 8 smartphones, and *iv*) 4 Fog devices and 16 smartphones.

5.2 Performance Evaluation

In this section, the environment of Fog-enabled CoT remote patient monitoring system is simulated. Then, the efficiency of three placement strategies (Cloud-only, Fog-default, and Fog-proposed) as shown in Fig. 5.1 are evaluated in terms of energy consumption, latency, and network usage.The network links for RPM application and the main simulation parameters are described in Table 5.1 and Table 5.2, respectively.

Table 5.1: Description of network links for RPM application

Source	Destination	Latency (ms)
Blood glucose (BG)	Smartphone	6
Smartphone	Wi-Fi gateway	2
Wi-Fi gateway	ISP gateway	4
ISP gateway	Cloud DC	100

Table 5.2: Main simulation parameter values

Parameter	Value
Simulation time	400 seconds
Cloud energy usage	150 - 400 W
Fog energy usage	70 - 130 W
Data sensing interval	5 ms

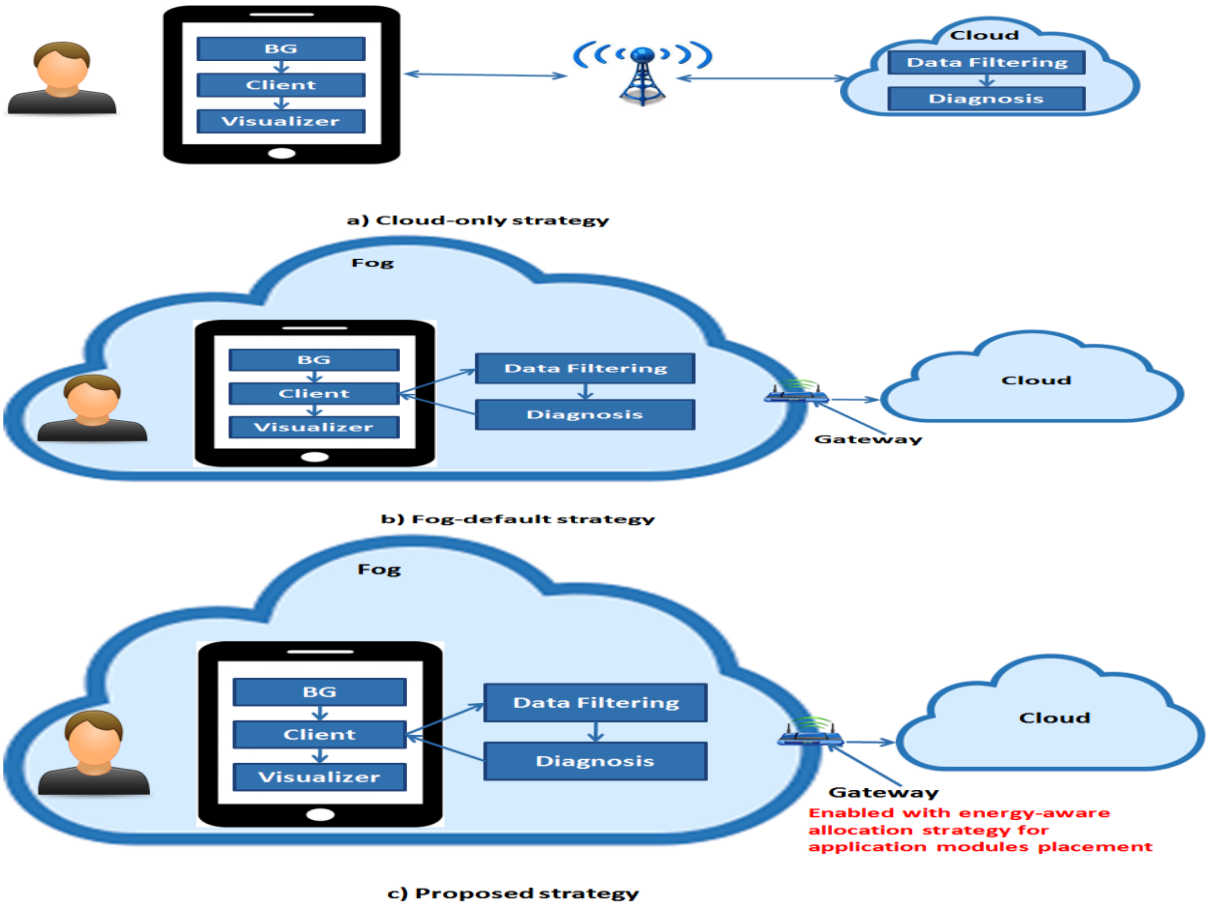

Figure 5.1. Application modules placements according to a) Cloud-only, b) Fog-default, and c) Fog-proposed strategies.

5.2.1 Energy Consumption

The energy consumed by the mobile devices, the Fog devices, and the Cloud datacenter is measured under three different strategies (the proposed allocation policy, the default policy, and Cloud-only policy). As shown in Figure 5.2, the proposed policy reduces the energy used by Fog devices by 8.27% compared to the default policy (already considered in iFogSim). The total energy consumed by various devices of the RPM scenario considering the proposed policy, the default policy, and without Fog integration is depicted in Figure 5.3. The proposed policy is observed to be more energy-efficient than the other two policies, saving approximately 2.72% of the energy compared to Cloud-only and 1.61% of the energy compared to the Fog-default.

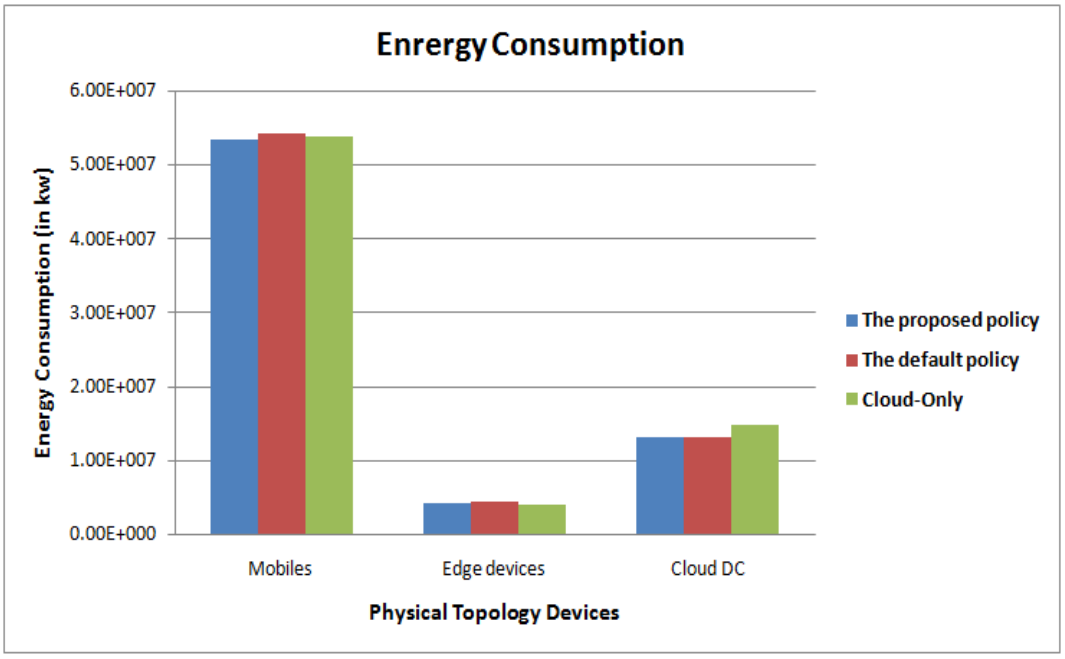

Figure 5.2. The energy consumed by different devices (mobile phones, edge devices, and the Cloud) under various policies (the proposed policy, the default policy and the Cloud-only policy).

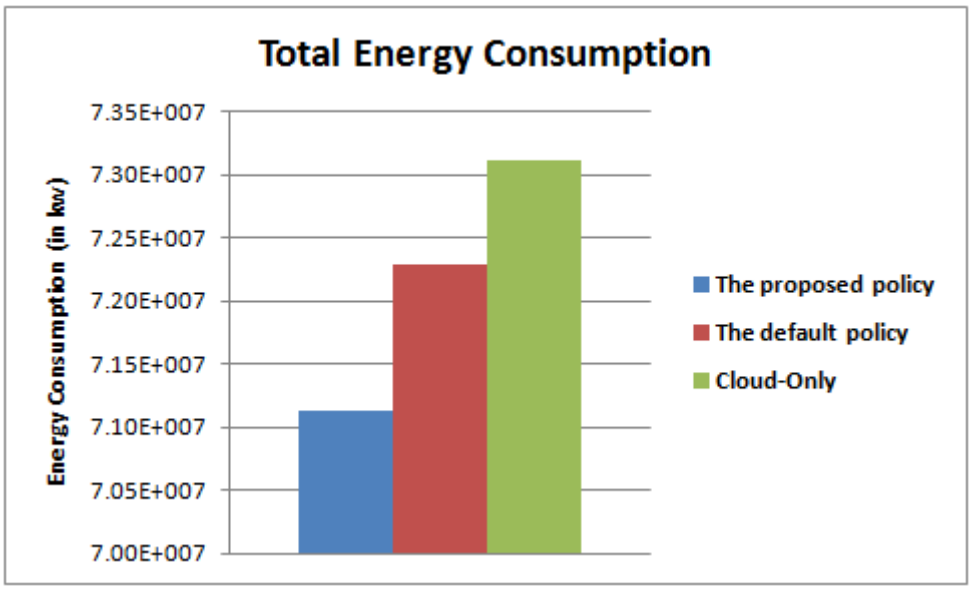

Figure 5.3. The total energy consumed in the RPM scenario under various allocation policies (the proposed policy, the default policy, and the Cloud-only policy).

5.2.2 End-to-End Latency Average

In the RPM scenario we consider, the end-to-end latency is represented by the loop from collecting the blood glucose state to displaying the results on the patient's smartphone, i.e., the client graphical user interface (GUI). It is reasonable that a high or unpredictable latency negatively affects the patient's health and the quality of service. As shown in Figure5.4, the use of Fog devices for processing can significantly reduce the latency compared to processing at a Cloud datacenter. Compared to the default allocation policy of application modules, the proposed policy has a lower latency, particularly in configurations three and four, as shown in Figure5.4.

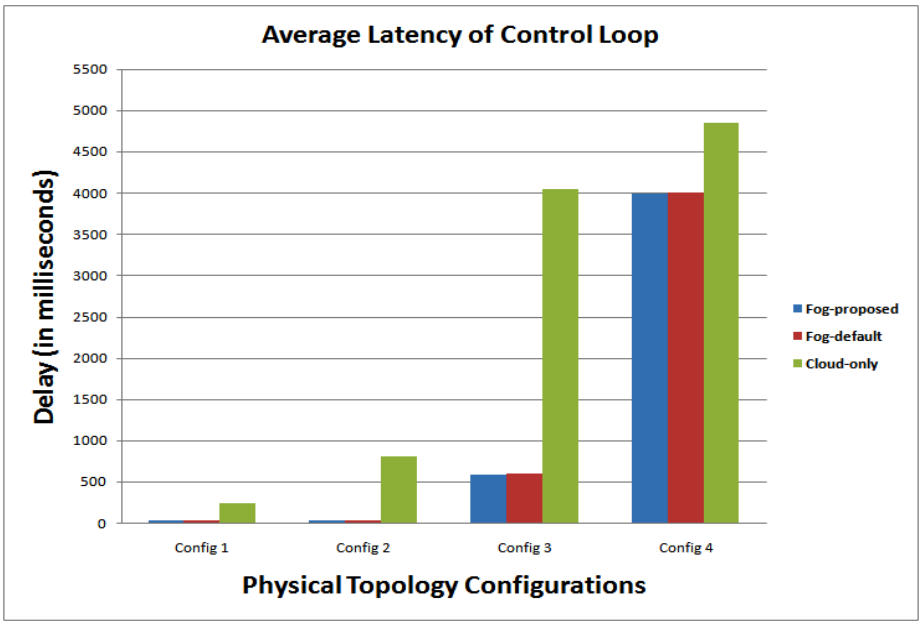

Figure 5.4. The end-to-end average latency of the RPM application loop under various allocation policies.

5.2.3 Network Usage

In general, the use of Fog computing reduces the amount of data transmitted over the network. This reduction arises from allocating most application modules to the network edge, avoiding the need to communicate with the Cloud. Therefore, a Fog-enabled CoT application scenario results in lower network usage than the two-tier CoT. Figure5.5illustrates the network usage, showing that the default allocation policy is slightly better than the proposed policy in configurations *i*) and *ii*), while the proposed policy is better in configurations *iii*) and *iv*) (i.e., heavy workloads).

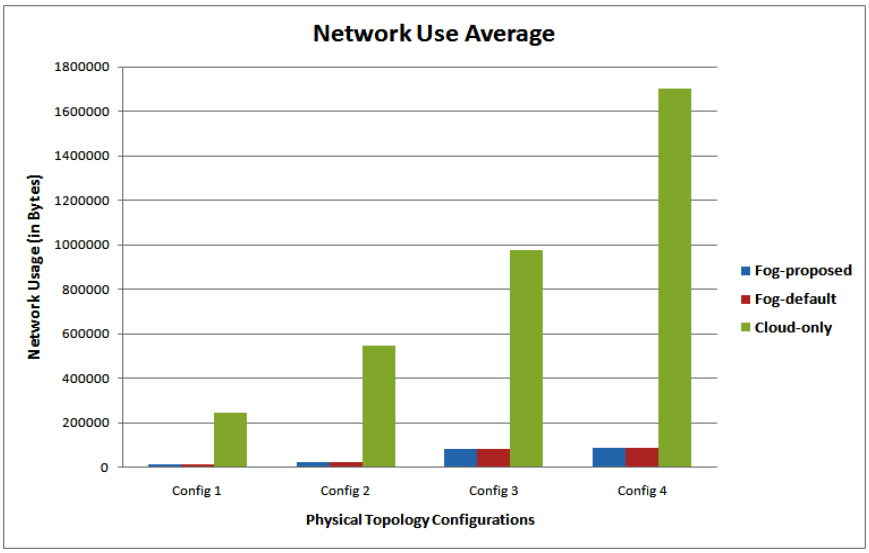

Figure 5.5.The network usage of the RPM application under various allocation policies.

5.3Summary

This chapter evaluates the performance of the proposed Model along with the energy-aware allocation algorithm compared to the edge-ward placement and the default allocation policies. The results showed that the proposed policy is more energy-efficient.

CHAPTER VI

CONCLUSION AND FUTURE WORK

6.1 Overview

This chapter summarizes the contributions and findings of the research work and presents suggestions and recommendations for future study. This research primarily investigates the importance of energy efficiency for achieving efficient healthcare services.

6.2 The Proposed Method

The proposed method to tackle the energy efficiency in CoT-based healthcare by allocating application modules in the energy-efficient Fog devices rather than the inefficient ones. It also balances the load between Fog devices which helps in achieving improved performance. The aim of this research is to propose energy-aware Fog-enabled Cloud of Things model for healthcare.

6.3 Contribution of the Research

As mentioned above in the previous section. The main goal of this thesis is to propose an energy-aware Fog-enabled Cloud of Things model for healthcare sector. The highlight of this thesis is that Fog layer was added, between the IoT and Cloud tiers, to reduce the data transmission and communication of IoT devices with the Cloud which significantly reduces the energy consumed. After, to achieve more energy efficiency, the thesis propose a strategy for allocating application modules in balanced and energy efficient manner. Therefore, this study reaches a number of contributions for energy efficiency especially on

healthcare. Major activities corresponding to contributions are summarized as follows:

i. **Identify theimportance of energy efficiency in gaining efficient and delay-sensitive healthcare services/applications.**

The importance of energy efficiency is identified by investigating and analyzing more than 25 recent proposals. Surveying the state of the art in this issue especially in the context of healthcare was helped successfully to propose new model for energy efficient CoT-based healthcare model.

ii. **Design an energy-aware Fog-enabled CoT model for healthcare depending on the results obtained on the first step (i).**

subsequent to conducting a comprehensive literature review and related work to same area of energy-efficient healthcare we have proposed a three-tier CoT model to be used in this thesis. Also, an algorithm for allocating application modules on Fog devices is proposed. To evaluate the efficiency of the proposed model, a comparison with Cloud-only and the Fog-default strategies in iFogsim simulator is done. The proposed model saving approximately 8% of the energy consumed at Fog devices. The simulation results showed that the proposed model is more energy efficient than the Cloud-only and the Fog-default.

6.4 FutureWork

The thesis achieved all the objectives of the study by determining the significant factors that affect energy efficiency in Cloud of things model such as data transmission and heavy communication with the Cloud.Based on these factors, a new Fog-enabled CoT model is proposed in order to obtain improved CoT-based healthcare services.However, several research opportunities still exist and further research can be conducted into them.

The future work will be achieved as the following objectives:

- ✓ To study the significance of energy efficiency in patient's mobility scenarios.
- ✓ To identify the role of Fog computing in support improved healthcare services in terms of latency, energy efficiency, and privacy with considering different healthcare deployment scenarios.
- ✓ To study the significance of smart gateways (Fog gateways) in performing sophisticated analytics on patients health records without violating regulations.

6.5 Summary

This chapter presents the summaries of the proposed model that has been done in this research. This study also proposes strategy for allocation of application modules in Fog devices which believed to affect the performance of Fog-enabled CoT-based healthcare in terms of energy consumption, latency, and network bandwidth. The results showed that the research objectives have been achieved. Furthermore, this research discusses the plan for future work to improve and extend the current work and how it will be applied to another CoT applications such as smart cities.

REFERENCES

1. Aazam, M. and Huh, E. N. (2014) ‘Fog computing and smart gateway based communication for cloud of things’, *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014*, pp. 464–470. doi: 10.1109/FiCloud.2014.83.
2. Aazam, M., Khan, I., Alsaffar, A. A. and Huh, E. (2014) ‘Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved’, in *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences & Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014*. IEEE, pp. 414–419. doi: 10.1109/IBCAST.2014.6778179.
3. Abedin, S. F., Alam, M. G. R., Haw, R. and Hong, C. S. (2015) ‘A system model for energy efficient green-IoT network’, in *International Conference on Information Networking*, pp. 177–182. doi: 10.1109/ICOIN.2015.7057878.
4. Ahmad, M., Amin, M. B., Hussain, S., Kang, B. H., Cheong, T. and Lee, S. (2016) ‘Health Fog: a novel framework for health and wellness applications’, *Journal of Supercomputing*. Springer US, 72(10), pp. 1–19. doi: 10.1007/s11227-016-1634-x.
5. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. and Ayyash, M. (2015) ‘Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications’, *IEEE*

Communications Surveys and Tutorials, 17(4), pp. 2347–2376. doi: 10.1109/COMST.2015.2444095.

6. Alduais, N. A. M. ., Abdullah, J., Jamil, A. and Audah, L. (2016) 'An Efficient Data Collection and Dissemination for IOT based WSN', in *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual. IEEE*, pp. 1–6.
7. Alnowiser, A., Aldhahri, E., Alahmadi, A. and Zhu, M. M. (2014) 'Enhanced Weighted Round Robin (EWRR) with DVFS Technology in Cloud Energy-Aware', in *2014 International Conference on Computational Science and Computational Intelligence*. IEEE, pp. 320–326. doi: 10.1109/CSCI.2014.62.
8. Aloi, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W. and Savaglio, C. (2017) 'Enabling IoT interoperability through opportunistic smartphone-based mobile gateways', *Journal of Network and Computer Applications*. Elsevier, 81(October), pp. 74–84. doi: 10.1016/j.jnca.2016.10.013.
9. Alskaif, T., Guerrero Zapata, M. and Bellalta, B. (2015) 'Game theory for energy efficiency in Wireless Sensor Networks: Latest trends', *Journal of Network and Computer Applications*, pp. 33–61. doi: 10.1016/j.jnca.2015.03.011.
10. Alvarez, R. C. (2002) 'The promise of e-Health - a Canadian perspective.', *Ehealth international*, 1(1), p. 4. doi: 10.1186/1476-3591-1-4.
11. Andriopoulou, F., Dagiuklas, T. and Orphanoudakis, T.

(2017) ‘Integrating IoT and Fog Computing for Healthcare Service Delivery’, in Keramidas, G., Voros, N., and Hübner, M. (eds) *Components and Services for IoT Platforms, Springer International Publishing Switzerland*. Cham: Springer International Publishing, pp. 213–232. doi: 10.1007/978-3-319-42304-3.

12. Avancha, S., Baxi, A. and Kotz, D. (2012) ‘Privacy in mobile technology for personal healthcare’, *ACM Computing Surveys (CSUR)*, 45(1), pp. 1–56. doi: 10.1145/0000000.0000000.

13. Babu, S. M., Lakshmi, A. J. and Rao, B. T. (2015) ‘A study on cloud based Internet of Things: CloudIoT’, in *2015 Global Conference on Communication Technologies (GCCT)*. IEEE, pp. 60–65. doi: 10.1109/GCCT.2015.7342624.

14. Baker, T., Asim, M., Tawfik, H., Aldawsari, B. and Buyya, R. (2017) ‘An energy-aware service composition algorithm for multiple cloud-based IoT applications’, *Journal of Network and Computer Applications*. Elsevier Ltd, 89, pp. 96–108. doi: 10.1016/j.jnca.2017.03.008.

15. Barcelo, M., Correa, A., Llorca, J., Tulino, A., Lopez Vicario, J. and Morell, A. (2016) ‘IoT-Cloud Service Optimization in Next Generation Smart Environments’, *IEEE Journal on Selected Areas in Communications*, 8716(c), pp. 1–1. doi: 10.1109/JSAC.2016.2621398.

16. Belli, L., Cirani, S., Davoli, L., Melegari, L., Mónton, A.

and Picone, M. (2015) ‘An Open-Source Cloud Architecture for Big Stream IoT Applications’, *Interoperability and Open-Source Solutions for the Internet of Things. Springer International Publishing*, pp. 73–88. doi: 10.1007/978-3-319-16546-2.

17. Benharref, A. and Serhani, M. A. (2014) ‘Novel cloud and SOA-based framework for E-health monitoring using wireless biosensors’, *IEEE Journal of Biomedical and Health Informatics*, 18(1), pp. 46–55. doi: 10.1109/JBHI.2013.2262659.

18. Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F. and Parashar, M. (2017) ‘Mobility-Aware Application Scheduling in Fog Computing’, *IEEE Cloud Computing*, 4(2), pp. 26–35. doi: 10.1109/MCC.2017.27.

19. Botta, A., De Donato, W., Persico, V. and Pescape, A. (2014) ‘On the integration of cloud computing and internet of things’, *Proceedings - 2014 International Conference on Future Internet of Things and Cloud, FiCloud 2014*, pp. 23–30. doi: 10.1109/FiCloud.2014.14.

20. Botta, A., de Donato, W., Persico, V. and Pescapé, A. (2016) ‘Integration of Cloud computing and Internet of Things: A survey’, *Future Generation Computer Systems*. Elsevier B.V., 56, pp. 684–700. doi: 10.1016/j.future.2015.09.021.

21. Botta, A., De Donato, W., Persico, V. and Pescapé, A. (2016) ‘Integration of Cloud computing and Internet of Things:

A survey', *Future Generation Computer Systems*, 56, pp. 684–700. doi: 10.1016/j.future.2015.09.021.

22. C. S., M. and N. K, S. (2015) 'Internet of Things Challengies and Apportunities', *Smart Sensors, Measurement and Instrumentation*, 42(3), pp. 287–307. doi: 10.1201/9781420052824.ch13.

23. Catarinucci, L., De Donno, D., Mainetti, L., Palano, L., Patrono, L., Stefanizzi, M. L. and Tarricone, L. (2015) 'An IoT-Aware Architecture for Smart Healthcare Systems', *IEEE Internet of Things Journal*, 2(6), pp. 515–526. doi: 10.1109/JIOT.2015.2417684.

24. Cavalcante, E., Pereira, J., Alves, M. P., Maia, P., Moura, R., Batista, T., Delicato, F. C. and Pires, P. F. (2016a) 'On the interplay of Internet of Things and Cloud Computing: A systematic mapping study', *Computer Communications*. Elsevier B.V., 89–90, pp. 17–33. doi: 10.1016/j.comcom.2016.03.012.

25. Cavalcante, E., Pereira, J., Alves, M. P., Maia, P., Moura, R., Batista, T., Delicato, F. C. and Pires, P. F. (2016b) 'On the interplay of Internet of Things and Cloud Computing: A systematic mapping study', *Computer Communications*. Elsevier B.V., 89–90, pp. 17–33. doi: 10.1016/j.comcom.2016.03.012.

26. Chang, J. Y. (2014) 'A Distributed Cluster Computing Energy-Efficient Routing Scheme for Internet of Things

Systems', *Wireless Personal Communications*. Springer US, pp. 757–776. doi: 10.1007/s11277-014-2251-8.

27. Chiang, M. and Zhang, T. (2016) 'Fog and IoT: An Overview of Research Opportunities', *IEEE Internet of Things Journal*, 3(6), pp. 854–864. doi: 10.1109/JIOT.2016.2584538.

28. Cisco Systems (2016) 'Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are', *Www.Cisco.Com*, p. 6.

29. Consortium, O. and Working, A. (2017) 'OpenFog Reference Architecture for Fog Computing', (February), pp. 1–162.

30. Dabbagh, M., Hamdaoui, B., Guizani, M. and Rayes, A. (2015) 'Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment', *IEEE Network*, 29(2), pp. 56–61. doi: 10.1109/MNET.2015.7064904.

31. Deng, R., Lu, R., Lai, C., Luan, T. H. and Liang, H. (2016) 'Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption', *IEEE Internet of Things Journal*, 3(6), pp. 1171–1181. doi: 10.1109/JIOT.2016.2565516.

32. Díaz, M., Martín, C. and Rubio, B. (2016a) '$$\lambda $$-CoAP: An Internet of Things and Cloud Computing Integration Based on the Lambda Architecture and CoAP', in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*. *Springer International*

Publishing, pp. 195–206. doi: 10.1007/978-3-319-28910-6_18.

33. Díaz, M., Martín, C. and Rubio, B. (2016b) 'State-of-the-art, challenges, and open issues in the integration of Internet of things and cloud computing', *Journal of Network and Computer Applications*. Elsevier, 67, pp. 99–117. doi: 10.1016/j.jnca.2016.01.010.

34. Distefano, S., Merlino, G. and Puliafito, A. (2012) 'Enabling the cloud of things', *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*, pp. 858–863. doi: 10.1109/IMIS.2012.61.

35. Doukas, C. and Maglogiannis, I. (2012) 'Bringing IoT and cloud computing towards pervasive healthcare', *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012*, pp. 922–926. doi: 10.1109/IMIS.2012.26.

36. Duan, H., Chen, C., Min, G. and Wu, Y. (2016) 'Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems', *Future Generation Computer Systems*, pp. 1–25. doi: 10.1016/j.future.2016.02.016.

37. Dubey, H., Yang, J., Constant, N., Amiri, A. M., Yang, Q. and Makodiya, K. (2015) 'Fog Data: Enhancing Telehealth Big Data Through Fog Computing', *Proceedings of the ASE BigData & SocialInformatics 2015*, p. 14:1--14:6. doi: 10.1145/2818869.2818889.

38. Etelapera, M., Vecchio, M. and Giaffreda, R. (2014) ‘Improving energy efficiency in IoT with re-configurable virtual objects’, in *2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, pp. 520–525. doi: 10.1109/WF-IoT.2014.6803222.

39. Etemad, M., Aazam, M. and St-Hilaire, M. (2017) ‘Using DEVS for modeling and simulating a Fog Computing environment’, in *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, pp. 849–854. doi: 10.1109/ICCNC.2017.7876242.

40. Fang, Y., Wang, F. and Ge, J. (2010) ‘A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing’, *In: Wang F.L., Gong Z., Luo X., Lei J. (eds) Web Information Systems and Mining. WISM 2010. Lecture Notes in Computer Science*. Springer, 6318, pp. 271–277. doi: 10.1007/978-3-642-16515-3_34.

41. Fortino, G., Di Fatta, G., Pathan, M. and Vasilakos, A. V. (2014) ‘Cloud-assisted body area networks: state-of-the-art and future challenges’, *Wireless Networks*, 20(7), pp. 1925–1938. doi: 10.1007/s11276-014-0714-1.

42. Fortino, G., Giannantonio, R., Gravina, R., Kuryloski, P. and Jafari, R. (2013) ‘Enabling effective programming and flexible management of efficient body sensor network applications’, *IEEE Transactions on Human-Machine Systems*, 43(1), pp. 115–133. doi: 10.1109/TSMCC.2012.2215852.

43. Fortino, G., Guerrieri, A., Russo, W. and Savaglio, C.

(2014) ‘Integration of agent-based and Cloud Computing for the smart objects-oriented IoT’, in *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, pp. 493–498. doi: 10.1109/CSCWD.2014.6846894.

44. Fortino, G., Parisi, D., Pirrone, V. and Di Fatta, G. (2014) ‘BodyCloud: A SaaS approach for community Body Sensor Networks’, *Future Generation Computer Systems*, 35, pp. 62–79. doi: 10.1016/j.future.2013.12.015.

45. Fortino, G., Savaglio, C., Palau, C. E., de Puga, J. S., Ganzha, M., Paprzycki, M., Montesinos, M., Liotta, A. and Llop, M. (2018) ‘Towards Multi-layer Interoperability of Heterogeneous IoT Platforms: The INTER-IoT Approach’, in, pp. 199–232. doi: 10.1007/978-3-319-61300-0_10.

46. Fratu, O., Pena, C., Craciunescu, R. and Halunga, S. (2015) ‘Fog computing system for monitoring Mild Dementia and COPD patients - Romanian case study’, *2015 12th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services, TELSIKS 2015*, pp. 123–128. doi: 10.1109/TELSKS.2015.7357752.

47. Gachet, D., De Buenaga, M., Aparicio, F. and Padr??n, V. (2012) ‘Integrating internet of things and cloud computing for health services provisioning: The virtual cloud carer project’, *Proceedings - 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS*

2012, pp. 918–921. doi: 10.1109/IMIS.2012.25.

48. Gia, T. N., Jiang, M., Rahmani, A.-M., Westerlund, T., Liljeberg, P. and Tenhunen, H. (2015) 'Fog Computing in Healthcare Internet of Things: A Case Study on ECG Feature Extraction', in *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*. IEEE, pp. 356–363. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.51.

49. Gia, T. N., Thanigaivelan, N. K., Rahmani, A. M., Westerlund, T., Liljeberg, P. and Tenhunen, H. (2015) 'Customizing 6LoWPAN networks towards Internet-of-Things based ubiquitous healthcare systems', in *NORCHIP 2014 - 32nd NORCHIP Conference: The Nordic Microelectronics Event*, pp. 1–6. doi: 10.1109/NORCHIP.2014.7004716.

50. Granados, J., Rahmani, A.-M., Nikander, P., Liljeberg, P. and Tenhunen, H. (2014) 'Towards Energy-Efficient HealthCare: an Internet-of-Things Architecture Using Intelligent Gateways', in *Proceedings of the 4th International Conference on Wireless Mobile Communication and Healthcare - 'Transforming healthcare through innovations in mobile and wireless technologies'*. ICST, pp. 279–282. doi: 10.4108/icst.mobihealth.2014.257394.

51. Gravina, R., Ma, C., Pace, P., Aloi, G., Russo, W., Li, W. and Fortino, G. (2017) 'Cloud-based Activity-aaService cyber–

physical framework for human activity monitoring in mobility', *Future Generation Computer Systems*. Elsevier B.V., 75, pp. 158–171. doi: 10.1016/j.future.2016.09.006.

52. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. and Buyya, R. (2017) 'iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments', *Software: Practice and Experience*, pp. 1–22. doi: 10.1002/spe.2509.

53. Hamdi, O., Chalouf, M. A., Ouattara, D. and Krief, F. (2014) 'EHealth: Survey on research projects, comparative study of telemonitoring architectures and main issues', *Journal of Network and Computer Applications*. Elsevier, 46, pp. 100–112. doi: 10.1016/j.jnca.2014.07.026.

54. Hans, L., Hr, Sadeghi, A.-R. and Winandy, M. (2010) 'Securing the e-health cloud', *Proceedings of the 1st ACM International Health Informatics Symposium*, pp. 220–229. doi: 10.1145/1882992.1883024.

55. Hassanalieragh, M., Page, A., Soyata, T., Sharma, G., Aktas, M., Mateos, G., Kantarci, B. and Andreescu, S. (2015) 'Health Monitoring and Management Using Internet-of-Things (IoT) Sensing with Cloud-Based Processing: Opportunities and Challenges', *Proceedings - 2015 IEEE International Conference on Services Computing, SCC 2015*, pp. 285–292. doi: 10.1109/SCC.2015.47.

56. Hou, L., Zhao, S., Xiong, X., Zheng, K., Chatzimisios, P.,

Hossain, M. S. and Xiang, W. (2016) ‘Internet of Things Cloud: Architecture and Implementation’, *IEEE Communications Magazine*, 54(12), pp. 32–39. doi: 10.1109/MCOM.2016.1600398CM.

57. Huang, J., Meng, Y., Gong, X., Liu, Y. and Duan, Q. (2014) ‘A Novel Deployment Scheme for Green Internet of Things’, *IEEE Internet of Things Journal*, PP(99), pp. 1–1. doi: 10.1109/JIOT.2014.2301819.

58. Ben Ida, I., Jemai, A. and Loukil, A. (2016) ‘A survey on security of IoT in the context of eHealth and clouds’, in *2016 11th International Design & Test Symposium (IDT)*. IEEE, pp. 25–30. doi: 10.1109/IDT.2016.7843009.

59. Jalali, F., Hinton, K., Ayre, R., Alpcan, T. and Tucker, R. S. (2016) ‘Fog computing may help to save energy in cloud computing’, *IEEE Journal on Selected Areas in Communications*, 34(5), pp. 1728–1739. doi: 10.1109/JSAC.2016.2545559.

60. Jalali, F., Vishwanath, A., de Hoog, J. and Suits, F. (2016) ‘Interconnecting Fog computing and microgrids for greening IoT’, in *2016 IEEE Innovative Smart Grid Technologies - Asia (ISGT-Asia)*. IEEE, pp. 693–698. doi: 10.1109/ISGT-Asia.2016.7796469.

61. Jara, a J., Zamora-Izquierdo, M. a and Skarmeta, a F. (2013) ‘Interconnection Framework for mHealth and Remote Monitoring Based on the Internet of Things’, *Ieee Journal on*

Selected Areas in Communications, 31(9), pp. 47–65. doi: 10.1109/jsac.2013.sup.0513005.

62. Jara, A. J., Fernández, D., López, P., Zamora, M. A., Marin, L. and Skarmeta, A. F. G. (2012) 'YOAPY: A Data Aggregation and Pre-processing Module for Enabling Continuous Healthcare Monitoring in the Internet of Things', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 248–255. doi: 10.1007/978-3-642-35395-6_34.

63. Kaur, K., Dhand, T., Kumar, N. and Zeadally, S. (2017) 'Container-as-a-Service at the Edge: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers', *IEEE Wireless Communications*, 24(3), pp. 48–56. doi: 10.1109/MWC.2017.1600427.

64. Kaur, N. and Sood, S. K. (2015) 'An Energy-Efficient Architecture for the Internet of Things (IoT)', *IEEE Systems Journal*, pp. 1–10. doi: 10.1109/JSYST.2015.2469676.

65. Khodadadi, F., A.V., D. and R., B. (2016) *Internet of things principles and paradigms*. Elsevier.

66. Kim, S. H. and Kim, D. (2015) 'Enabling Multi-Tenancy via Middleware-Level Virtualization with Organization Management in the Cloud of Things', *IEEE Transactions on Services Computing*, 8(6), pp. 971–984. doi: 10.1109/TSC.2014.2355828.

67. Koubaa, A. and Shakshuki, E. (2015) ‘Cloud of Things: Integration of IoT with Cloud Computing’, *Robots and Sensor Clouds*, 36, pp. 77–94. doi: 10.1007/978-3-319-22168-7.

68. Kraemer, F. A., Braten, A. E., Tamkittikhun, N. and Palma, D. (2017) ‘Fog Computing in Healthcare–A Review and Discussion’, *IEEE Access*, 5, pp. 9206–9222. doi: 10.1109/ACCESS.2017.2704100.

69. Li, S., Xu, L. Da and Zhao, S. (2015) ‘The internet of things: a survey’, *Information Systems Frontiers*, 17(2), pp. 243–259. doi: 10.1007/s10796-014-9492-7.

70. Li, W., Santos, I., Delicato, F. C., Pires, P. F., Pirmez, L., Wei, W., Song, H., Zomaya, A. and Khan, S. (2017) ‘System modelling and performance evaluation of a three-tier Cloud of Things’, *Future Generation Computer Systems*. Elsevier B.V., 70, pp. 104–125. doi: 10.1016/j.future.2016.06.019.

71. Lu, R., Lin, X. and Shen, X. (2013) ‘SPOC: A Secure and Privacy-Preserving Opportunistic Computing Framework for Mobile-Healthcare Emergency’, *IEEE Transactions on Parallel and Distributed Systems*, 24(3), pp. 614–624. doi: 10.1109/TPDS.2012.146.

72. Luo, S. and Ren, B. (2016) ‘The monitoring and managing application of cloud computing based on Internet of Things’, *Computer Methods and Programs in Biomedicine*. Elsevier Ireland Ltd, 130, pp. 154–161. doi: 10.1016/j.cmpb.2016.03.024.

73. Mangali, N. K. and Kota, V. K. (2015) ‘Health monitoring systems: An energy efficient data collection technique in wireless sensor networks’, in *2015 International Conference on Microwave, Optical and Communication Engineering (ICMOCE)*. IEEE, pp. 130–133. doi: 10.1109/ICMOCE.2015.7489707.

74. Mao, Y., Chengfa, L., Guihai, C. and Wu, J. (2005) ‘EECS: an energy efficient clustering scheme in wireless sensor networks’, *Performance Computing and Communications Conference 2005 IPCCC 2005 24th IEEE International*, 3(November), pp. 535–540. doi: 10.1109/PCCC.2005.1460630.

75. Martinovic, G. and Zoric, B. (2012) ‘E-health framework based on autonomic cloud computing’, *Proceedings - 2nd International Conference on Cloud and Green Computing and 2nd International Conference on Social Computing and Its Applications, CGC/SCA 2012*, pp. 214–218. doi: 10.1109/CGC.2012.36.

76. Masip-Bruin, X., Marín-Tordera, E., Tashakor, G., Jukan, A. and Ren, G.-J. (2016) ‘Foggy clouds and cloudy fogs: a real need for coordinated management of fog-to-cloud computing systems’, *IEEE Wireless Communications*, 23(5), pp. 120–128. doi: 10.1109/MWC.2016.7721750.

77. Mehmood, A., Khan, S., Shams, B. and Lloret, J. (2015) ‘Energy-efficient multi-level and distance-aware clustering mechanism for WSNs’, *International Journal of*

Communication Systems, 28(5), pp. 972–989. doi: 10.1002/dac.2720.

78. Mohammed, J., Lung, C.-H., Ocneanu, A., Thakral, A., Jones, C. and Adler, A. (2014) 'Internet of Things: Remote Patient Monitoring Using Web Services and Cloud Computing', *2014 IEEE International Conference on Internet of Things(iThings), and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom)*, (iThings), pp. 256–263. doi: 10.1109/iThings.2014.45.

79. Mohsen Nia, A., Mozaffari-Kermani, M., Sur-Kolay, S., Raghunathan, A. and Jha, N. (2015) 'Energy-Efficient Long-term Continuous Personal Health Monitoring', *IEEE Transactions on Multi-Scale Computing Systems*, 1(2), pp. 1–1. doi: 10.1109/TMSCS.2015.2494021.

80. Nan, Y., Li, W., Bao, W., Delicato, F. C., Pires, P. F. and Zomaya, A. Y. (2016) 'Cost-effective processing for Delay-sensitive applications in Cloud of Things systems', in *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*. IEEE, pp. 162–169. doi: 10.1109/NCA.2016.7778612.

81. Oueis, J., Strinati, E. C. and Barbarossa, S. (2015) 'The Fog Balancing: Load Distribution for Small Cell Cloud Computing', in *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, pp. 1–6. doi:

10.1109/VTCSpring.2015.7146129.

82. Pandya, H. B. and Champaneria, T. A. (2015) ‘Internet of things: Survey and case studies’, *International Conference on Electrical, Electronics, Signals, Communication and Optimization, EESCO 2015*. doi: 10.1109/EESCO.2015.7253713.

83. Papageorgiou, A., Zahn, M. and Kovacs, E. (2014) ‘Efficient auto-configuration of energy-related parameters in cloud-based IoT platforms’, *2014 IEEE 3rd International Conference on Cloud Networking, CloudNet 2014*, pp. 236–241. doi: 10.1109/CloudNet.2014.6968998.

84. Perera, C., Zaslavsky, A., Christen, P. and Georgakopoulos, D. (2014) ‘Context aware computing for the internet of things: A survey’, *IEEE Communications Surveys and Tutorials*, 16(1), pp. 414–454. doi: 10.1109/SURV.2013.042313.00197.

85. Petrolo, R., Morabito, R., Loscrì, V. and Mitton, N. (2016) ‘The design of the gateway for the Cloud of Things’, *Annals of Telecommunications*. doi: 10.1007/s12243-016-0521-z.

86. Pradhan, P., Behera, P. K. and Ray, B. N. B. (2016) ‘Modified Round Robin Algorithm for Resource Allocation in Cloud Computing’, in *Procedia Computer Science*. Elsevier Masson SAS, pp. 878–890. doi: 10.1016/j.procs.2016.05.278.

87. Prasad, S. S. and Ieee, S. M. (2012) ‘An Energy Efficient and Reliable Internet of Things’, *In Communication,*

Information & Computing Technology (ICCICT), 2012 International Conference on IEEE, pp. 1–4.

88. Prasad, S. S. and Kumar, C. (2013) ‘A Green and Reliable Internet of Things’, *Communications and Network*, 05(01), pp. 44–48. doi: 10.4236/cn.2013.51B011.

89. Praveena, N. G. and Prabha, H. (2014) ‘An efficient multi-level clustering approach for a heterogeneous wireless sensor network using link correlation’, *EURASIP Journal on Wireless Communications and Networking*, 2014(1), pp. 1–10. doi: 10.1186/1687-1499-2014-168.

90. Ramalho, F., Neto, A., Santos, K., Filho, J. B. and Agoulmine, N. (2016) ‘Enhancing eHealth smart applications: A Fog-enabled approach’, in *2015 17th International Conference on E-Health Networking, Application and Services, HealthCom 2015*, pp. 323–328. doi: 10.1109/HealthCom.2015.7454519.

91. Rani, S., Talwar, R., Malhotra, J., Ahmed, S., Sarkar, M. and Song, H. (2015) ‘A Novel Scheme for an Energy Efficient Internet of Things Based on Wireless Sensor Networks’, *Sensors*, 15(11), pp. 28603–28626. doi: 10.3390/s151128603.

92. Rao, B. B. P., Saluia, P., Sharma, N., Mittal, a and Sharma, S. V (2012) ‘Cloud computing for Internet of Things & sensing based applications’, *Sensing Technology (ICST), 2012 Sixth International Conference on*, pp. 374–380. doi: 10.1109/ICSensT.2012.6461705.

93. Rault, T., Bouabdallah, A., Challal, Y. and Marin, F.

(2014) ‘Energy-efficient architecture for wearable sensor networks’, in *2014 IFIP Wireless Days (WD)*. IEEE, pp. 1–8. doi: 10.1109/WD.2014.7020803.

94. Riazul Islam, S. M., Daehan Kwak, Humaun Kabir, M., Hossain, M. and Kyung-Sup Kwak (2015) ‘The Internet of Things for Health Care: A Comprehensive Survey’, *IEEE Access*, 3, pp. 678–708. doi: 10.1109/ACCESS.2015.2437951.

95. Rohokale, V. M., Prasad, N. R. and Prasad, R. (2011) ‘A cooperative Internet of Things (IoT) for rural healthcare monitoring and control’, *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace and Electronic Systems Technology, Wireless VITAE 2011*. doi: 10.1109/WIRELESSVITAE.2011.5940920.

96. Sahu, Y., Pateriya, R. K. and Gupta, R. K. (2013) ‘Cloud Server Optimization with Load Balancing and Green Computing Techniques Using Dynamic Compare and Balance Algorithm’, in *2013 5th International Conference on Computational Intelligence and Communication Networks*. IEEE, pp. 527–531. doi: 10.1109/CICN.2013.114.

97. Saied, Y. Ben, Olivereau, A., Zeghlache, D. and Laurent, M. (2014) ‘Lightweight collaborative key establishment scheme for the Internet of Things’, *Computer Networks*. Elsevier B.V., 64, pp. 273–295. doi: 10.1016/j.comnet.2014.02.001.

98. Sarkar, S., Chatterjee, S. and Misra, S. (2015) ‘Assessment

of the Suitability of Fog Computing in the Context of Internet of Things', *IEEE Transactions on Cloud Computing*, PP(99), pp. 1–1. doi: 10.1109/TCC.2015.2485206.

99. Sawand, A., Djahel, S., Zhang, Z. and Nait-Abdesselam, F. (2015) 'Multidisciplinary approaches to achieving efficient and trustworthy eHealth monitoring systems', *2014 IEEE/CIC International Conference on Communications in China, ICCC 2014*, pp. 187–192. doi: 10.1109/ICCChina.2014.7008269.

100. Sebestyen, G., Hangan, A., Oniga, S. and Gal, Z. (2014) 'eHealth solutions in the context of internet of things', *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014*. doi: 10.1109/AQTR.2014.6857876.

101. Shah, T., Yavari, A., Mitra, K., Saguna, S., Jayaraman, P. P., Rabhi, F. and Ranjan, R. (2016) 'Remote health care cyber-physical system: quality of service (QoS) challenges and opportunities', *IET Cyber-Physical Systems: Theory & Applications*, 1(1), pp. 40–48. doi: 10.1049/iet-cps.2016.0023.

102. Shi, Y., Ding, G., Wang, H., Eduardo Roman, H. and Lu, S. (2015) 'The fog computing service for healthcare', *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare, Ubi-HealthTech 2015*, pp. 70–74. doi: 10.1109/Ubi-HealthTech.2015.7203325.

103. Shi, Y., Ding, G., Wang, H., Roman, H. E. and Lu, S.

(2015) ‘The fog computing service for healthcare’, in *2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech)*. IEEE, pp. 1–5. doi: 10.1109/Ubi-HealthTech.2015.7203325.

104. Singh, S., Swaroop, A., Kumar, A. and Anamika (2016) ‘A survey on techniques to achive energy efficiency in cloud computing’, in *2016 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, pp. 1281–1285. doi: 10.1109/CCAA.2016.7813915.

105. Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J. P., Riahi, M., Aberer, K., Jayaraman, P. P., Zaslavsky, A., ??arko, I. P., Skorin-Kapov, L. and Herzog, R. (2015) ‘OpenIoT: Open source internet-of-things in the cloud’, in *Interoperability and Open-Source Solutions for the Internet of Things. Springer International Publishing*, pp. 13–25. doi: 10.1007/978-3-319-16546-2_3.

106. Souza, V. B. C., Ramirez, W., Masip-Bruin, X., Marin-Tordera, E., Ren, G. and Tashakor, G. (2016) ‘Handling service allocation in combined Fog-cloud scenarios’, in *2016 IEEE International Conference on Communications (ICC)*. IEEE, pp. 1–5. doi: 10.1109/ICC.2016.7511465.

107. Souza, V. B., Masip-Bruin, X., Marin-Tordera, E., Ramirez, W. and Sanchez, S. (2016) ‘Towards Distributed Service Allocation in Fog-to-Cloud (F2C) Scenarios’, in *2016*

IEEE Global Communications Conference (GLOBECOM). IEEE, pp. 1–6. doi: 10.1109/GLOCOM.2016.7842341.

108. Stojmenovic, I. (2015) 'Fog computing: A cloud to the ground support for smart things and machine-to-machine networks', in *2014 Australasian Telecommunication Networks and Applications Conference, ATNAC 2014*, pp. 117–122. doi: 10.1109/ATNAC.2014.7020884.

109. Suciu, G., Suciu, V. and Fratu, O. (2013) 'Big Data Processing for E-Health Applications using a Decentralized Cloud M2M System', *Latest Trends on Systems*, II, pp. 232–237.

110. Tang, J., Zhou, Z., Niu, J. and Wang, Q. (2014) 'An energy efficient hierarchical clustering index tree for facilitating time-correlated region queries in the Internet of Things', *Journal of Network and Computer Applications*. Elsevier, 40(1), pp. 1–11. doi: 10.1016/j.jnca.2013.07.009.

111. Verma, Sagar and Yadav, Aran Kumar and Motwani, Deepak and Raw, RS and Singh, H. K. (2016) 'An efficient data replication and load balancing technique for fog computing environment', in *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on, IEEE*. IEEE, pp. 2888–2895.

112. Wang, S., Huang, X., Liu, Y. and Yu, R. (2016) 'CachinMobile: An energy-efficient users caching scheme for fog computing', in *2016 IEEE/CIC International Conference on*

Communications in China (ICCC). IEEE, pp. 1–6. doi: 10.1109/ICCChina.2016.7636852.

113. Wang, Y. H. and Wu, I. C. (2009) 'Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms', *Software - Practice and Experience*, 39(7), pp. 701–736. doi: 10.1002/spe.

114. Wankhade, N. R. and Choudhari, D. N. (2016) 'Novel Energy Efficient Election Based Routing Algorithm for Wireless Sensor Network', *Procedia Computer Science*. Elsevier Masson SAS, 79, pp. 772–780. doi: 10.1016/j.procs.2016.03.101.

115. Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J. and Rovatsos, M. (2017) 'Fog Orchestration for Internet of Things Services', *IEEE Internet Computing*, 21(2), pp. 16–24. doi: 10.1109/MIC.2017.36.

116. Whitmore, A., Agarwal, A. and Da Xu, L. (2015) 'The Internet of Things???A survey of topics and trends', *Information Systems Frontiers*, 17(2), pp. 261–274. doi: 10.1007/s10796-014-9489-2.

117. Xuan-Qui Pham and Eui-Nam Huh (2016) 'Towards task scheduling in a cloud-fog computing system', in *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, pp. 1–4. doi: 10.1109/APNOMS.2016.7737240.

118. Yaacoub, E., Kadri, A. and Abu-Dayya, A. (2012)

‘Cooperative wireless sensor networks for green internet of things’, in *Proceedings of the 8h ACM symposium on QoS and security for wireless and mobile networks*, p. 79. doi: 10.1145/2387218.2387235.

119. Yi, S., Li, C. and Li, Q. (2015) ‘A Survey of Fog Computing’, in *Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata ’15*. New York, New York, USA: ACM Press, pp. 37–42. doi: 10.1145/2757384.2757397.

120. You, L., Liu, C. and Tong, S. (2011) ‘Community Medical Network (CMN): Architecture and implementation’, *2011 Global Mobile Congress, GMC 2011*. doi: 10.1109/GMC.2011.6103930.

121. Yu, S., Kim, J. and Lee, J. (2013) ‘Lifetime improvement method using mobile sink for IoT service’, in *PE-WASUN 2013 - Proceedings of the 10th ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, Co-located with ACM MSWiM 2013*, pp. 145–149. doi: 10.1145/2507248.2507273.

122. Zhang, Y., Ren, J., Liu, J., Xu, C., Guo, H. and Liu, Y. (2017) ‘A Survey on Emerging Computing Paradigms for Big Data’, *Chinese Journal of Electronics*, 26(1), pp. 1–12. doi: 10.1049/cje.2016.11.016.

123. Zhou, J., Leppanen, T., Harjula, E., Ylianttila, M., Ojala, T., Yu, C. and Jin, H. (2013) ‘CloudThings: A common architecture for integrating the Internet of Things with Cloud

Computing', *Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design, CSCWD 2013*, pp. 651–657. doi: 10.1109/CSCWD.2013.6581037.

124. Zhu, N., Diethe, T., Camplani, M., Tao, L., Burrows, A., Twomey, N., Kaleshi, D., Mirmehdi, M., Flach, P. and Craddock, I. (2015) 'Bridging e-Health and the Internet of Things: The SPHERE Project', *IEEE Intelligent Systems*, 30(4), pp. 39–46.

Appendix A

Use Case (Remote Patient Monitoring) Simulation Code:

```
package org.fog.test.perfeval;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.HashMap;

import java.util.LinkedList;

import java.util.List;


import org.cloudbus.cloudsim.DatacenterBroker;

import org.cloudbus.cloudsim.Host;

import org.cloudbus.cloudsim.Log;

import org.cloudbus.cloudsim.Pe;

import org.cloudbus.cloudsim.Storage;

import org.cloudbus.cloudsim.VmSchedulerTimeShared;

import org.cloudbus.cloudsim.core.CloudSim;

import
org.cloudbus.cloudsim.power.PowerVmAllocationPolicySingleThres
hold;

import org.cloudbus.cloudsim.power.PowerHost;

import org.cloudbus.cloudsim.power.PowerVmSelectionPolicy;

import
org.cloudbus.cloudsim.power.PowerVmSelectionPolicyMinimumUtili
zation;

import
org.cloudbus.cloudsim.power.models.PowerModelSpecPower_BAZAR;

import org.cloudbus.cloudsim.provisioners.BwProvisionerSimple;

import org.cloudbus.cloudsim.provisioners.PeProvisionerSimple;
```

```
import
org.cloudbus.cloudsim.provisioners.RamProvisionerSimple;

import
org.cloudbus.cloudsim.sdn.overbooking.BwProvisionerOverbooking
;

import
org.cloudbus.cloudsim.sdn.overbooking.PeProvisionerOverbooking
;

import org.cloudbus.cloudsim.xml.CloudletDatas;

import org.cloudbus.cloudsim.xml.DatacenterDatas;

import org.cloudbus.cloudsim.xml.DvfsDatas;

import org.cloudbus.cloudsim.xml.HostDatas;

import org.cloudbus.cloudsim.xml.SimulationXMLParse;

import org.cloudbus.cloudsim.xml.VmDatas;

import org.fog.application.AppEdge;

import org.fog.application.AppLoop;

import org.fog.application.Application;

import org.fog.application.selectivity.FractionalSelectivity;

import org.fog.entities.Actuator;

import org.fog.entities.FogBroker;

import org.fog.entities.FogDevice;

import org.fog.entities.FogDeviceCharacteristics;

import org.fog.entities.Sensor;

import org.fog.entities.Tuple;

import org.fog.placement.Controller;

import org.fog.placement.ModuleMapping;

import org.fog.placement.ModulePlacementEdgewards;

import org.fog.placement.ModulePlacementMapping;

import org.fog.policy.AppModuleAllocationPolicy;
```

```
import org.fog.policy.AppModuleAllocationPolicyWRR;
import org.fog.scheduler.StreamOperatorScheduler;
import org.fog.utils.FogLinearPowerModel;
import org.fog.utils.FogLinearPowerModelDVFS;
import org.fog.utils.FogUtils;
import org.fog.utils.TimeKeeper;
import org.fog.utils.distribution.DeterministicDistribution;

/**
 * Simulation setup for case study 1 - BG Beam Tractor Game
 * @author Harshit Gupta
 *
 */
public class RPM22 {
	static List<FogDevice> fogDevices = new
ArrayList<FogDevice>();
	static List<Sensor> sensors = new ArrayList<Sensor>();
	static List<Actuator> actuators = new
ArrayList<Actuator>();

	static boolean CLOUD = false;

	static int numOfDepts = 2; //4
	static int numOfMobilesPerDept = 4; //Number of mobile
users
	static double BG_TRANSMISSION_TIME = 5.1;
	static double GC_TRANSMISSION_TIME = 5.1;
```

```
    //static double BG_TRANSMISSION_TIME = 10;

    //added variables
    /*
    private static int DCNumber;
    private static int hostsNumber;
    private static int vmsTotalNumber;
    private static int no_cur_vm=0;

    private static int cloudletsTotalNumber;
    private static int no_cur_cloudlet=0;

    private static ArrayList<DatacenterBroker> vect_dcbroker
;

    private static ArrayList<DatacenterDatas> vect_dcs ;
    private static ArrayList<HostDatas> vect_hosts ;
    private static ArrayList<VmDatas> vect_vms ;
    private static ArrayList<CloudletDatas> vect_cls ;
    private static DvfsDatas ConfigDvfs;
    private static SimulationXMLParse ConfSimu;
    */
    public static void main(String[] args) {

        Log.printLine("Starting RPM System...");
        /*vect_dcs = new ArrayList<>();
        vect_hosts = new ArrayList<>();
```

```
            vect_vms = new ArrayList<>();
            vect_cls = new ArrayList<>();*/
            try {
                Log.disable();
                /* Configuration Variables*/
                /* XML configuration file Parsing*/
                /*ConfSimu = new
SimulationXMLParse(System.getProperty("user.dir")+"/Experience
.xml");
                vect_dcs = ConfSimu.getArrayListDCS();
                vect_hosts =
vect_dcs.get(0).getArrayListHosts();
                vect_vms = ConfSimu.getArrayListVMS();
                vect_cls = ConfSimu.getArrayListCLS();

                DCNumber = vect_dcs.size();
                cloudletsTotalNumber = vect_cls.size();
                hostsNumber = vect_hosts.size();
                vmsTotalNumber = vect_vms.size();*/

                int num_user = 1; // number of cloud users
                Calendar calendar = Calendar.getInstance();
                boolean trace_flag = false; // mean trace
events

                CloudSim.init(num_user, calendar, trace_flag);

                String appId = "rpm"; // identifier of the
application
```

```
                String appId2 = "rpm2"; // identifier of the
application


                FogBroker broker = new FogBroker("broker");

                FogBroker broker2 = new FogBroker("broker2");


                Application application =
createApplication(appId, broker.getId());

                Application application2 =
createApplication(appId2, broker2.getId());


                application.setUserId(broker.getId());

                application2.setUserId(broker2.getId());


                createFogDevices(broker.getId(), appId);

                createFogDevices2(broker2.getId(), appId2);



                ModuleMapping moduleMapping =
ModuleMapping.createModuleMapping(); // initializing a module
mapping

                //App1


                if(CLOUD){

                     // if the mode of deployment is cloud-
based


     /*moduleMapping.addModuleToDevice("client_GUI", "cloud",
numOfDepts*numOfMobilesPerDept); // fixing all instances of
the client_GUI module to the Cloud
```

```
     moduleMapping.addModuleToDevice("diagnostic_calculator",
"cloud", numOfDepts*numOfMobilesPerDept); // fixing all
instances of the DIAGNOSIS_RESULT Calculator module to the
Cloud

*/
     //moduleMapping.addModuleToDevice("client_GUI", "cloud");
// fixing all instances of the client_GUI module to the Cloud


     moduleMapping.addModuleToDevice("diagnostic_calculator",
"cloud"); // fixing all instances of the DIAGNOSIS_RESULT
Calculator module to the Cloud



               }else{

                    // if the mode of deployment is cloud-
based

                    //.addModuleToDevice("client_GUI",
"cloud", numOfDepts*numOfMobilesPerDept); // fixing all
instances of the client_GUI module to the Cloud


     //moduleMapping.addModuleToDevice("diagnostic_calculator"
, "cloud"); // fixing all instances of the client_GUI module
to the Cloud

                    // rest of the modules will be placed by
the Edge-ward placement policy



                    for(FogDevice device : fogDevices){

                         if(device.getName().startsWith("m")){


     //moduleMapping.addModuleToDevice("client",
device.getName(), 1);  // fixing all instances of the Client
module to the Smartphones


     moduleMapping.addModuleToDevice("client_GUI",
device.getName());  // fixing all instances of the Client
module to the Smartphones
```

```
                        }
                        if(device.getName().startsWith("d")){


    //moduleMapping.addModuleToDevice("client",
device.getName(), 1);  // fixing all instances of the Client
module to the Smartphones


    moduleMapping.addModuleToDevice("diagnostic_calculator",
device.getName());  // fixing all instances of the Client
module to the Smartphones




                        }
                    }
                }


          //App2
               if(CLOUD){
                    // if the mode of deployment is cloud-
based


    /*moduleMapping.addModuleToDevice("client_GUI", "cloud",
numOfDepts*numOfMobilesPerDept); // fixing all instances of
the client_GUI module to the Cloud


    moduleMapping.addModuleToDevice("diagnostic_calculator",
"cloud", numOfDepts*numOfMobilesPerDept); // fixing all
instances of the DIAGNOSIS_RESULT Calculator module to the
Cloud
*/
    //moduleMapping.addModuleToDevice("client_GUI", "cloud");
// fixing all instances of the client_GUI module to the Cloud
```

```
     moduleMapping.addModuleToDevice("diagnostic_calculator2",
"cloud"); // fixing all instances of the DIAGNOSIS_RESULT
Calculator module to the Cloud


               }else{

                    // if the mode of deployment is cloud-
based

                    //.addModuleToDevice("client_GUI",
"cloud", numOfDepts*numOfMobilesPerDept); // fixing all
instances of the client_GUI module to the Cloud


     //moduleMapping.addModuleToDevice("diagnostic_calculator"
, "cloud"); // fixing all instances of the client_GUI module
to the Cloud

                    // rest of the modules will be placed by
the Edge-ward placement policy


                    for(FogDevice device : fogDevices){


     if(device.getName().startsWith("m2")){


     //moduleMapping.addModuleToDevice("client",
device.getName(), 1);  // fixing all instances of the Client
module to the Smartphones


     moduleMapping.addModuleToDevice("client_GUI2",
device.getName());  // fixing all instances of the Client
module to the Smartphones

                         }


     if(device.getName().startsWith("d2")){



     //moduleMapping.addModuleToDevice("client",
```

```
device.getName(), 1);  // fixing all instances of the Client
module to the Smartphones


     moduleMapping.addModuleToDevice("diagnostic_calculator2",
device.getName());  // fixing all instances of the Client
module to the Smartphones




                          }

                     }

                }

                Controller controller = new Controller("master-
controller", fogDevices, sensors,

                          actuators);



                controller.submitApplication(application, 0,

                          (CLOUD)?(new
ModulePlacementMapping(fogDevices, application,
moduleMapping))

                                    :(new
ModulePlacementEdgewards(fogDevices, sensors, actuators,
application, moduleMapping)));

                controller.submitApplication(application2, 0,

                          (CLOUD)?(new
ModulePlacementMapping(fogDevices, application2,
moduleMapping))

                                    :(new
ModulePlacementEdgewards(fogDevices, sensors, actuators,
application2, moduleMapping)));


     TimeKeeper.getInstance().setSimulationStartTime(Calendar.
getInstance().getTimeInMillis());
```

```
                CloudSim.startSimulation();


                CloudSim.stopSimulation();



                Log.printLine("RPM System finished!");
            } catch (Exception e) {
                e.printStackTrace();
                Log.printLine("Unwanted errors happen");
            }
        }


        /**
         * Creates the fog devices in the physical topology of
the simulation.
         * @param userId
         * @param appId
         */
        private static void createFogDevices(int userId, String
appId) {
            FogDevice cloud = createFogDevice2("cloud", 44800,
40000, 100, 10000, 0, 0.01, 16*103, 16*83.25); // creates the
fog device Cloud at the apex of the hierarchy with level=0
            cloud.setParentId(-1);
            FogDevice proxy = createFogDevice2("proxy-server",
10000, 4000, 10000, 10000, 1, 0.0, 107.339, 83.4333); //
creates the fog device Proxy Server (level=1)
            proxy.setParentId(cloud.getId()); // setting Cloud
as parent of the Proxy Server
```

```
          proxy.setUplinkLatency(100); // latency of
connection from Proxy Server to the Cloud is 100 ms


          fogDevices.add(cloud);

          fogDevices.add(proxy);


          for(int i=0;i<numOfDepts;i++){

               addGw(i+"", userId, appId, proxy.getId()); //
adding a fog device for every Gateway in physical topology.
The parent of each gateway is the Proxy Server

          }


     }


     private static void createFogDevices2(int userId, String
appId2) {

          FogDevice cloud = createFogDevice2("cloud", 44800,
40000, 100, 10000, 0, 0.01, 16*103, 16*83.25); // creates the
fog device Cloud at the apex of the hierarchy with level=0

          cloud.setParentId(-1);

          FogDevice proxy = createFogDevice2("proxy-server",
10000, 4000, 10000, 10000, 1, 0.0, 107.339, 83.4333); //
creates the fog device Proxy Server (level=1)

          proxy.setParentId(cloud.getId()); // setting Cloud
as parent of the Proxy Server

          proxy.setUplinkLatency(100); // latency of
connection from Proxy Server to the Cloud is 100 ms


          fogDevices.add(cloud);

          fogDevices.add(proxy);
```

```
          for(int i=0;i<numOfDepts;i++){

               addGw2(i+"", userId, appId2, proxy.getId()); //
adding a fog device for every Gateway in physical topology.
The parent of each gateway is the Proxy Server

          }


     }


     private static FogDevice addGw(String id, int userId,
String appId, int parentId){

          FogDevice dept = createFogDevice("d-"+id, 10000,
4000, 10000, 10000, 1, 0.0, 107.339, 83.4333);

          fogDevices.add(dept);

          dept.setParentId(parentId);

          dept.setUplinkLatency(4); // latency of connection
between gateways and proxy server is 4 ms

          for(int i=0;i<numOfMobilesPerDept;i++){

               String mobileId = id+"-"+i;

               FogDevice mobile = addMobile(mobileId, userId,
appId, dept.getId()); // adding mobiles to the physical
topology. Smartphones have been modeled as fog devices as
well.

               mobile.setUplinkLatency(2); // latency of
connection between the smartphone and proxy server is 2 ms

               fogDevices.add(mobile);

          }

          return dept;

     }


     private static FogDevice addGw2(String id, int userId,
String appId2, int parentId){
```

```
        FogDevice dept = createFogDevice("d2-"+id, 10000,
4000, 10000, 10000, 1, 0.0, 107.339, 83.4333);

        fogDevices.add(dept);

        dept.setParentId(parentId);

        dept.setUplinkLatency(4); // latency of connection
between gateways and proxy server is 4 ms

        for(int i=0;i<numOfMobilesPerDept;i++){

            String mobileId = id+"-"+i;

            FogDevice mobile = addMobile2(mobileId, userId,
appId2, dept.getId()); // adding mobiles to the physical
topology. Smartphones have been modeled as fog devices as
well.

            mobile.setUplinkLatency(2); // latency of
connection between the smartphone and proxy server is 2 ms

            fogDevices.add(mobile);

        }

        return dept;

    }

    private static FogDevice addMobile(String id, int userId,
String appId, int parentId){

        FogDevice mobile = createFogDevice2("m-"+id, 1000,
1000, 10000, 270, 3, 0, 87.53, 82.44);

        mobile.setParentId(parentId);

        Sensor bgSensor = new Sensor("s-"+id, "BG", userId,
appId, new DeterministicDistribution(BG_TRANSMISSION_TIME));
// inter-transmission time of BG sensor follows a
deterministic distribution

        sensors.add(bgSensor);

        Actuator display = new Actuator("a-"+id, userId,
appId, "DISPLAY");

        Actuator alertActuater = new Actuator("a-"+id,
userId, appId, "ALERT_CALL");
```

```
        actuators.add(display);

        actuators.add(alertActuater);

        bgSensor.setGatewayDeviceId(mobile.getId());

        bgSensor.setLatency(6.0);  // latency of connection
between BG sensors and the parent Smartphone is 6 ms

        display.setGatewayDeviceId(mobile.getId());

        alertActuater.setGatewayDeviceId(mobile.getId());

        display.setLatency(1.0);  // latency of connection
between Display actuator and the parent Smartphone is 1 ms

        alertActuater.setLatency(1.0);

        return mobile;

    }

    private static FogDevice addMobile2(String id, int
userId, String appId2, int parentId){

        FogDevice mobile = createFogDevice2("m2-"+id, 1000,
1000, 10000, 270, 3, 0, 87.53, 82.44);

        mobile.setParentId(parentId);

        Sensor gcSensor = new Sensor("s2-"+id, "GC", userId,
appId2, new DeterministicDistribution(GC_TRANSMISSION_TIME));
// inter-transmission time of BG sensor follows a
deterministic distribution

        sensors.add(gcSensor);

        Actuator display = new Actuator("a2-"+id, userId,
appId2, "DISPLAY");

        Actuator alertActuater = new Actuator("a2-"+id,
userId, appId2, "ALERT_CALL");

        actuators.add(display);

        actuators.add(alertActuater);

        gcSensor.setGatewayDeviceId(mobile.getId());

        gcSensor.setLatency(6.0);

        display.setGatewayDeviceId(mobile.getId());
```

```
        alertActuater.setGatewayDeviceId(mobile.getId());
        display.setLatency(1.0);  // latency of connection
between Display actuator and the parent Smartphone is 1 ms
        alertActuater.setLatency(1.0);
        return mobile;
    }

    /**
     * Creates a vanilla fog device
     * @param nodeName name of the device to be used in
simulation
     * @param mips MIPS
     * @param ram RAM
     * @param upBw uplink bandwidth
     * @param downBw downlink bandwidth
     * @param level hierarchy level of the device
     * @param ratePerMips cost rate per MIPS used
     * @param busyPower
     * @param idlePower
     * @return
     */
    private static FogDevice createFogDevice(String nodeName,
long mips,
            int ram, long upBw, long downBw, int level,
double ratePerMips, double busyPower, double idlePower) {

        List<Pe> peList = new ArrayList<Pe>();

        // 3. Create PEs and add these into a list.
```

```
          peList.add(new Pe(0, new
PeProvisionerOverbooking(mips))); // need to store Pe id and
MIPS Rating

          // added three Pe to be Quad-core

          peList.add(new Pe(1, new
PeProvisionerOverbooking(mips)));

          peList.add(new Pe(2, new
PeProvisionerOverbooking(mips)));

          peList.add(new Pe(3, new
PeProvisionerOverbooking(mips)));

          int hostId = FogUtils.generateEntityId();

          long storage = 1000000; // host storage

          int bw = 10000;

          double maxPower;

          double staticPowerPercent;



          List<Host> hostList = new ArrayList<Host>();

          //added code

          boolean enableDVFS; // is the Dvfs enable on the
host

          ArrayList<Double>freqs ; // frequencies available by
the CPU

          HashMap<Integer,String> govs;  // Definition of Dvfs
Governor , and redefine specifics values


          /*

               HostDatas tmp_host = vect_hosts.get(0);


               ConfigDvfs = tmp_host.getDvfsDatas();
```

```
                maxPower = tmp_host.getMaxP();
                staticPowerPercent = tmp_host.getStaticPP();

                mips = tmp_host.getMips();
                ram = tmp_host.getRam();
                storage = tmp_host.getStorage();
                bw = tmp_host.getBw();
                freqs = tmp_host.getCpuFrequencies();
                govs = tmp_host.getHTGovs();
                enableDVFS = tmp_host.isDvfsEnable();

                //List<Pe> peList = new ArrayList<Pe>();

                int nb_pe = tmp_host.getCpus();



                for(int pe=0 ; pe < nb_pe ; pe++)
                {
                     peList.add(new Pe(pe, new
PeProvisionerSimple(mips),freqs,govs.get(pe), ConfigDvfs));
                }*/
          //peList.add(new Pe(0, new
PeProvisionerSimple(mips),null,null, null));
          PowerHost host = new PowerHost(
                     hostId,
                     new RamProvisionerSimple(ram),
                     new BwProvisionerOverbooking(bw),
```

```
                    storage,
                    peList,
                    new StreamOperatorScheduler(peList),
                    //new FogLinearPowerModel(busyPower,
idlePower)
                    //new FogLinearPowerModelDVFS(busyPower,
idlePower,peList)
                    new PowerModelSpecPower_BAZAR(peList)
                );
                hostList.add(host);



            String arch = "x86"; // system architecture
            String os = "Linux"; // operating system
            String vmm = "Xen";
            double time_zone = 10.0; // time zone this resource
located
            double cost = 3.0; // the cost of using processing
in this resource
            double costPerMem = 0.05; // the cost of using
memory in this resource
            double costPerStorage = 0.001; // the cost of using
storage in this
                                                    // resource
            double costPerBw = 0.0; // the cost of using bw in
this resource
            LinkedList<Storage> storageList = new
LinkedList<Storage>(); // we are not adding SAN


     // devices by now
```

```
          FogDeviceCharacteristics characteristics = new
FogDeviceCharacteristics(

                    arch, os, vmm, hostList, time_zone, cost,
costPerMem,

                    costPerStorage, costPerBw);


          FogDevice fogdevice = null;


          try {

               fogdevice = new FogDevice(nodeName,
characteristics,

                         new
AppModuleAllocationPolicy(hostList), storageList, 10, upBw,
downBw, 0, ratePerMips);

          } catch (Exception e) {

               e.printStackTrace();

          }


          fogdevice.setLevel(level);

          return fogdevice;

     }
// Create Cloud DC device method

     private static FogDevice createFogDevice2(String
nodeName, long mips,

               int ram, long upBw, long downBw, int level,
double ratePerMips, double busyPower, double idlePower) {


          List<Pe> peList = new ArrayList<Pe>();


          // 3. Create PEs and add these into a list.
```

```
          peList.add(new Pe(0, new
PeProvisionerOverbooking(mips))); // need to store Pe id and
MIPS Rating

          //added three Pe to be Quad-core

                    peList.add(new Pe(1, new
PeProvisionerOverbooking(mips)));

                    peList.add(new Pe(2, new
PeProvisionerOverbooking(mips)));

                    peList.add(new Pe(3, new
PeProvisionerOverbooking(mips)));


          int hostId = FogUtils.generateEntityId();

          long storage = 1000000; // host storage

          int bw = 10000;

          double maxPower;

          double staticPowerPercent;



          List<Host> hostList = new ArrayList<Host>();

          //added code

          boolean enableDVFS; // is the Dvfs enable on the
host

          ArrayList<Double>freqs ; // frequencies available by
the CPU

          HashMap<Integer,String> govs;  // Definition of Dvfs
Governor , and redefine specifics values


          /*

               HostDatas tmp_host = vect_hosts.get(0);
```

```
                ConfigDvfs = tmp_host.getDvfsDatas();

                maxPower = tmp_host.getMaxP();
                staticPowerPercent = tmp_host.getStaticPP();

                mips = tmp_host.getMips();
                ram = tmp_host.getRam();
                storage = tmp_host.getStorage();
                bw = tmp_host.getBw();
                freqs = tmp_host.getCpuFrequencies();
                govs = tmp_host.getHTGovs();
                enableDVFS = tmp_host.isDvfsEnable();

                //List<Pe> peList = new ArrayList<Pe>();

                int nb_pe = tmp_host.getCpus();



                for(int pe=0 ; pe < nb_pe ; pe++)
                {
                     peList.add(new Pe(pe, new
PeProvisionerSimple(mips),freqs,govs.get(pe), ConfigDvfs));
                }*/
          //peList.add(new Pe(0, new
PeProvisionerSimple(mips),null,null, null));
          PowerHost host = new PowerHost(
                     hostId,
                     new RamProvisionerSimple(ram),
```

```
                    new BwProvisionerOverbooking(bw),
                    storage,
                    peList,
                    new StreamOperatorScheduler(peList),
                    new FogLinearPowerModel(busyPower,
idlePower)
                    //new FogLinearPowerModelDVFS(busyPower,
idlePower,peList)
                    //new PowerModelSpecPower_BAZAR(peList)
                );
                hostList.add(host);



            String arch = "x86"; // system architecture
            String os = "Linux"; // operating system
            String vmm = "Xen";
            double time_zone = 10.0; // time zone this resource
located
            double cost = 3.0; // the cost of using processing
in this resource
            double costPerMem = 0.05; // the cost of using
memory in this resource
            double costPerStorage = 0.001; // the cost of using
storage in this
                                                    // resource
            double costPerBw = 0.0; // the cost of using bw in
this resource
            LinkedList<Storage> storageList = new
LinkedList<Storage>(); // we are not adding SAN


     // devices by now
```

```
          FogDeviceCharacteristics characteristics = new
FogDeviceCharacteristics(
                    arch, os, vmm, hostList, time_zone, cost,
costPerMem,
                    costPerStorage, costPerBw);


          FogDevice fogdevice = null;
          PowerVmSelectionPolicy powerVmSelectionPolicy = new
     PowerVmSelectionPolicyMinimumUtilization();
          try {
               fogdevice = new FogDevice(nodeName,
characteristics,
                         new
AppModuleAllocationPolicyWRR(hostList), storageList, 10, upBw,
downBw, 0, ratePerMips);
          } catch (Exception e) {
               e.printStackTrace();
          }


          fogdevice.setLevel(level);
          return fogdevice;
     }
// end of method
     /**
      * Function to create the BG Tractor Beam game
application in the DDF model.
      * @param appId unique identifier of the application
      * @param userId identifier of the user of the
application
```

```
     * @return

     */

    //@SuppressWarnings({"serial" })

    private static Application createApplication(String
appId, int userId){


        Application application =
Application.createApplication(appId, userId); // creates an
empty application model (empty directed graph)


        /*

         * Adding modules (vertices) to the application
model (directed graph)

         */

        //application.addAppModule("client", 10); // adding
module Client to the application model

        application.addAppModule("client_GUI", 10); //
adding module client_GUI to the application model

        application.addAppModule("diagnostic_calculator",
10); // adding module DIAGNOSIS_RESULT Calculator to the
application model


        /*

         * Connecting the application modules (vertices) in
the application model (directed graph) with edges

         */

        //if(BG_TRANSMISSION_TIME==10)

        //   application.addAppEdge("BG", "client_GUI",
2000, 500, "BG", Tuple.UP, AppEdge.SENSOR); // adding edge
from BG (sensor) to Client module carrying tuples of type BG

        //else
```

```
        application.addAppEdge("BG", "client_GUI", 3000,
500, "BG", Tuple.UP, AppEdge.SENSOR);

        application.addAppEdge("client_GUI",
"diagnostic_calculator", 3500, 500, "_SENSOR", Tuple.UP,
AppEdge.MODULE); // adding edge from Client to
DIAGNOSIS_RESULT Calculator module carrying tuples of type
_SENSOR

        application.addAppEdge("diagnostic_calculator",
"client_GUI", 100, 1000, "DIAGNOSIS_RESULT", Tuple.DOWN,
AppEdge.MODULE); // adding periodic edge (period=1000ms) from
DIAGNOSIS_RESULT Calculator to client_GUI module carrying
tuples of type PATIENT_STATE


        //application.addAppEdge("diagnostic_calculator",
"client", 14, 500, "DIAGNOSIS_RESULT", Tuple.DOWN,
AppEdge.MODULE);  // adding edge from DIAGNOSIS_RESULT
Calculator to Client module carrying tuples of type
DIAGNOSIS_RESULT

        //application.addAppEdge("client_GUI", "client",
100, 28, 1000, "GLOBAL_PATIENT_STATE", Tuple.DOWN,
AppEdge.MODULE); // adding periodic edge (period=1000ms) from
client_GUI to Client module carrying tuples of type
GLOBAL_PATIENT_STATE

        application.addAppEdge("client_GUI", "DISPLAY",
1000, 500, "SELF_STATE_UPDATE", Tuple.DOWN, AppEdge.ACTUATOR);
// adding edge from Client module to Display (actuator)
carrying tuples of type SELF_STATE_UPDATE


        /*

         * Defining the input-output relationships
(represented by selectivity) of the application modules.

         */

        application.addTupleMapping("client_GUI", "BG",
"_SENSOR", new FractionalSelectivity(0.9)); // 0.9 tuples of
type _SENSOR are emitted by Client module per incoming tuple
of type BG
```

```
        application.addTupleMapping("client_GUI",
"DIAGNOSIS_RESULT", "SELF_STATE_UPDATE", new
FractionalSelectivity(1.0)); // 1.0 tuples of type
SELF_STATE_UPDATE are emitted by Client module per incoming
tuple of type DIAGNOSIS_RESULT

        application.addTupleMapping("diagnostic_calculator",
"_SENSOR", "DIAGNOSIS_RESULT", new
FractionalSelectivity(1.0)); // 1.0 tuples of type
DIAGNOSIS_RESULT are emitted by DIAGNOSIS_RESULT Calculator
module per incoming tuple of type _SENSOR



        /*

         * Defining application loops to monitor the latency
of.

         * Here, we add only one loop for monitoring :
BG(sensor) -> Client -> DIAGNOSIS_RESULT Calculator -> Client
-> DISPLAY (actuator)

         */

        final AppLoop loop1 = new AppLoop(new
ArrayList<String>(){{add("BG");add("client_GUI");}});

        final AppLoop loop2 = new AppLoop(new
ArrayList<String>(){{add("client_GUI");add("diagnostic_calcula
tor");add("client_GUI");add("DISPLAY");}});

        List<AppLoop> loops = new
ArrayList<AppLoop>(){{add(loop1);add(loop2);}};

        application.setLoops(loops);



        return application;

    }

private static Application createApplication2(String appId2,
int userId){
```

```
          Application application2 =
Application.createApplication(appId2, userId); // creates an
empty application model (empty directed graph)



          /*

           * Adding modules (vertices) to the application
model (directed graph)

           */

          //application.addAppModule("client", 10); // adding
module Client to the application model

          application2.addAppModule("client_GUI2", 10); //
adding module client_GUI to the application model

          application2.addAppModule("diagnostic_calculator2",
10); // adding module DIAGNOSIS_RESULT Calculator to the
application model



          /*

           * Connecting the application modules (vertices) in
the application model (directed graph) with edges

           */

          //if(BG_TRANSMISSION_TIME==10)

          //   application.addAppEdge("BG", "client_GUI",
2000, 500, "BG", Tuple.UP, AppEdge.SENSOR); // adding edge
from BG (sensor) to Client module carrying tuples of type BG

          //else

          application2.addAppEdge("GC", "client_GUI", 3000,
500, "GC", Tuple.UP, AppEdge.SENSOR);

          application2.addAppEdge("client_GUI2",
"diagnostic_calculator2", 3500, 500, "_SENSOR", Tuple.UP,
AppEdge.MODULE); // adding edge from Client to
DIAGNOSIS_RESULT Calculator module carrying tuples of type
_SENSOR
```

```
          application2.addAppEdge("diagnostic_calculator2",
"client_GUI2", 100, 1000, "DIAGNOSIS_RESULT", Tuple.DOWN,
AppEdge.MODULE); // adding periodic edge (period=1000ms) from
DIAGNOSIS_RESULT Calculator to client_GUI module carrying
tuples of type PATIENT_STATE

          //application.addAppEdge("diagnostic_calculator",
"client", 14, 500, "DIAGNOSIS_RESULT", Tuple.DOWN,
AppEdge.MODULE);  // adding edge from DIAGNOSIS_RESULT
Calculator to Client module carrying tuples of type
DIAGNOSIS_RESULT

          //application.addAppEdge("client_GUI", "client",
100, 28, 1000, "GLOBAL_PATIENT_STATE", Tuple.DOWN,
AppEdge.MODULE); // adding periodic edge (period=1000ms) from
client_GUI to Client module carrying tuples of type
GLOBAL_PATIENT_STATE

          application2.addAppEdge("client_GUI2", "DISPLAY",
1000, 500, "SELF_STATE_UPDATE", Tuple.DOWN, AppEdge.ACTUATOR);
// adding edge from Client module to Display (actuator)
carrying tuples of type SELF_STATE_UPDATE



          /*

           * Defining the input-output relationships
(represented by selectivity) of the application modules.

           */

          application2.addTupleMapping("client_GUI2", "GC",
"_SENSOR", new FractionalSelectivity(0.9)); // 0.9 tuples of
type _SENSOR are emitted by Client module per incoming tuple
of type BG

          application2.addTupleMapping("client_GUI2",
"DIAGNOSIS_RESULT", "SELF_STATE_UPDATE", new
FractionalSelectivity(1.0)); // 1.0 tuples of type
SELF_STATE_UPDATE are emitted by Client module per incoming
tuple of type DIAGNOSIS_RESULT


     application2.addTupleMapping("diagnostic_calculator2",
"_SENSOR", "DIAGNOSIS_RESULT", new
FractionalSelectivity(1.0)); // 1.0 tuples of type
```

```
DIAGNOSIS_RESULT are emitted by DIAGNOSIS_RESULT Calculator
module per incoming tuple of type _SENSOR


          /*

           * Defining application loops to monitor the latency
of.

           * Here, we add only one loop for monitoring :
BG(sensor) -> Client -> DIAGNOSIS_RESULT Calculator -> Client
-> DISPLAY (actuator)

           */

          final AppLoop loop1 = new AppLoop(new
ArrayList<String>(){{add("GC");add("client_GUI2");}});

          final AppLoop loop2 = new AppLoop(new
ArrayList<String>(){{add("client_GUI2");add("diagnostic_calcul
ator2");add("client_GUI2");add("DISPLAY");}});

          List<AppLoop> loops = new
ArrayList<AppLoop>(){{add(loop1);add(loop2);}};

          application2.setLoops(loops);


          return application2;

     }

}
```

Appendix B

Application Module Allocation Algorithm Code:

```
package org.fog.policy;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.cloudbus.cloudsim.Host;
import org.cloudbus.cloudsim.Log;
import org.cloudbus.cloudsim.Vm;
import org.cloudbus.cloudsim.VmAllocationPolicy;
import org.cloudbus.cloudsim.core.CloudSim;
import org.cloudbus.cloudsim.power.PowerHost;
import org.cloudbus.cloudsim.power.PowerVmAllocationPolicyAbstract;

public class AppModuleAllocationPolicyWRR extends PowerVmAllocationPolicyAbstract {

	private final Map<String, Host> vm_table = new HashMap<String, Host>();

	private final CircularHostList hosts;

	public AppModuleAllocationPolicyWRR(List<? extends Host> list) {
		super(list);
		this.hosts = new CircularHostList(list);
```

```
}
//added method

        public PowerHost findHostForVm(Vm vm, CircularHostList hosts) {
                double minPower = Double.MAX_VALUE;
                PowerHost allocatedHost = null;

                for (PowerHost host : this.<PowerHost> getHostList()) {
                        if (hosts.contains(host)) {
                                continue;
                        }
                        if (host.isSuitableForVm(vm)) {

                                try {
                                        double powerAfterAllocation =
estimateConsumedEnergyAfterAllocation (host, vm);
                                        if (powerAfterAllocation != -1) {
                                                double powerDiff =
powerAfterAllocation - host.getPower();
                                                if (powerDiff < minPower) {
                                                        minPower = powerDiff;
                                                        allocatedHost = host;
                                                }
                                        }
                                } catch (Exception e) {
                                }
                        }
                }
```

```
                return allocatedHost;
        }
        /**
         * Gets the power after allocation.
         *
         * @param host the host
         * @param vm the vm
         *
         * @return the power after allocation
         */
        protected double estimateConsumedEnergyAfterAllocation (PowerHost host,
Vm vm) {
                double power = 0;
                try {
                        power =
host.getPowerModel().getPower(getMaxUtilizationAfterAllocation(host, vm));
                } catch (Exception e) {
                        e.printStackTrace();
                        System.exit(0);
                }
                return power;
        }
        protected double getMaxUtilizationAfterAllocation(PowerHost host, Vm vm)
{
                double requestedTotalMips = vm.getCurrentRequestedTotalMips();
                double hostUtilizationMips = host.getPreviousUtilizationMips();
                double hostPotentialUtilizationMips = hostUtilizationMips +
requestedTotalMips;
```

```
                double pePotentialUtilization = hostPotentialUtilizationMips /
host.getTotalMips();
                return pePotentialUtilization;
        }
    // end of added method
    @Override
    public boolean allocateHostForVm(Vm vm) {
        if (this.vm_table.containsKey(vm.getUid())) {
            return true;
        }

        boolean vm_allocated = false;

        Host host = this.hosts.next();
        if (host != null) {
            vm_allocated = this.allocateHostForVm(vm, host);
        }

        return vm_allocated;
    }

    @Override
    public boolean allocateHostForVm(Vm vm, Host host)
    {
        host = findHostForVm(vm, this.hosts);

        if (host != null && host.vmCreate(vm))
```

```
                {
                        vm_table.put(vm.getUid(), host);
                        Log.formatLine("%.4f: VM #" + vm.getId() + " has been allocated to
the host#" + host.getId() +
                                        " datacenter #" + host.getDatacenter().getId() + "(" +
host.getDatacenter().getName() + ") #",
                                        CloudSim.clock());
                        return true;
                }
                return false;
        }

        @Override
        public List<Map<String, Object>> optimizeAllocation(List<? extends Vm> vmList)
{
                return null;
        }

        @Override
        public void deallocateHostForVm(Vm vm) {
                Host host = this.vm_table.remove(vm.getUid());

                if (host != null) {
                        host.vmDestroy(vm);
                }
        }

        @Override
```

```
	public Host getHost(Vm vm) {
		return this.vm_table.get(vm.getUid());
	}

	@Override
	public Host getHost(int vmId, int userId) {
		return this.vm_table.get(Vm.getUid(userId, vmId));
	}


}
```