Quality of service and privacy in internet of things dedicated to healthcare

Randa Mohammed Abdelmonem Aboelfotoh

Quality of Service and Privacy in Internet of Things Dedicated to Healthcare

A Thesis Submitted to Faculty of Computer and Information Sciences - Ainshams University and University of Avignon
In Partial Fulfillment of the Requirements for The Dual Philosophy Degree in Computer and Information Sciences

By

Randa Mohamed Abdelmonem Aboelfotoh
Assistant Lecturer, Faculty of Computer and Information Sciences
Ainshams University, Egypt.

Supervisors

Prof. Dr. Eman Shabaan
Professor of computer systems
Faculty of Computer and Information Sciences
Ain-Shams University
Egypt

Prof. Dr. Abderrahim Benslimane
Professor in computer science
Laboratoire d'Informatique d'Avignon (LIA)
University of Avignon
France

2020

Acknowledgment

First of all, i must thank god for showing me the right way and providing me with patience and ability to introduce my thesis.

I would like to express my sincere gratitude to my advisor Prof. Eman Shaaban for her motivation, enthusiasm, wise guidance, and immense knowledge. I would also like to thank my advisor Prof. Abderrahim Benslimane for the continuous support of my Ph.D study and research.

Last but not least, i would like to thank my family who give me the support throughout all stages of my life with their deepest love and care, and supporting me during the compilation of this dissertation.

Abstract

The Internet of Things (IoT) based healthcare systems usually composed of medical and environmental sensors, remote servers, and the network. These systems focus on providing remote monitoring, disease diagnosis, and treatment progress observation. The healthcare systems in IoT domain helps in realizing long-term economical, ubiquitous, and patient centered care systems, that result in improving treatment and patient outcomes. This research contributes to the domain by proposing a Cloud-Fog based architecture that can embrace multiple healthcare scenarios, and able to adapt dynamically with the context and status of the patients. It allows the mobility and physical activity of the patients in the environment through deployment and implementation of an appropriate Received Signal Strength (RSS) based handoff mechanism. It also proposes a mobility-aware task scheduling and allocation approach in cloud-fog computing paradigm, called MobMBAR, with the objective of minimizing the total schedule time (makespan). MobMBAR performs dynamic balanced healthcare tasks distribution between the cloud and fog devices. It is a data locality based approach that depends on changing the location where the data is computed to where it actually resides. It takes scheduling decisions considering the priorities of tasks represented in their classifications and maximum response time. To evaluate the performance, we conduct an intensive simulation study with different stationery and mobility scenarios, and compare against other state of art solutions. We measure the performance metrics: makespan, network load, energy consumption, percentage of missed tasks, latency, execution cost, number of handoffs, and resource utilization, and study the effect of varying number of tasks, number of cloud devices, handoff threshold, and percentage of mobile devices on the performance metrics.

The experiments show acceptable results in terms of makespan, miss ratio, cost, latency, and network load. In case of mobility support, it shows that missed tasks ratios doesn't exceed one thousandths percent, and is proven to be 88% lower than state-of-the-art solutions in terms of makespan and 92% lower in terms of energy consumption. Our research also includes a realistic simulation case study that uses the layout of an indoor hospital building in Chicago, and it has demonstrated acceptable performance. To authenticate and secure communication between IoT device and gateways, the thesis also proposes a DTLS (Datagram Transport Layer Security) based mobility-enabled authentica-

tion scheme for IoT architecture. It ensures mutual authenticated handoff between mobile IoT devices and visited gateways while saving additional handshakes overhead. The performance of the proposed scheme is evaluated in terms of handshake time, processing time, energy consumption, and memory overhead. The results demonstrate its feasibility for limited resource devices.

Résumé

L'Internet des objets (IoT) est un système de santé basé sur des capteurs médicaux et environnementaux, des serveurs à distance et le réseau. Ces systèmes se concentrent sur la surveillance à distance, le diagnostic des maladies et l'observation de l'évolution des traitements. Les systèmes de soins de santé du domaine de l'IdO contribuent à la mise en place de systèmes de soins économiques, omniprésents et centrés sur le patient à long terme, qui permettent d'améliorer les traitements et les résultats pour les patients. Cette recherche contribue au domaine en proposant une architecture basée sur le principe du brouillard nuageux qui peut englober plusieurs scénarios de soins de santé et s'adapter de manière dynamique au contexte et à l'état des patients. Elle permet la mobilité et l'activité physique des patients dans l'environnement grâce au déploiement et à la mise en œuvre d'un mécanisme de transfert approprié basé sur la force du signal reçu (RSS). Il propose également une approche de planification et d'attribution des tâches tenant compte de la mobilité dans le cadre du paradigme de l'informatique dans le brouillard, appelée MobMBAR, dans le but de minimiser le temps total de planification (makepan). MobMBAR effectue une répartition dynamique et équilibrée des tâches de soins de santé entre les dispositifs de brouillard et de nuage. Il s'agit d'une approche basée sur la localisation des données qui dépend du changement de l'endroit où les données sont calculées à l'endroit où elles résident réellement. Elle prend des décisions de planification en tenant compte des priorités des tâches représentées dans leurs classifications et du temps de réponse maximum. Pour évaluer les performances, nous menons une étude de simulation intensive avec différents scénarios de papeterie et de mobilité, et nous les comparons à d'autres solutions de pointe. Nous mesurons les paramètres de performance : la capacité de production, la charge du réseau, la consommation d'énergie, le pourcentage de tâches manquées, le temps de latence, le coût d'exécution, le nombre de transferts et l'utilisation des ressources, et nous étudions l'effet d'un nombre variable de tâches, du nombre d'appareils en nuage, du seuil de transfert et du pourcentage d'appareils mobiles sur les paramètres de performance.

Les expériences montrent des résultats acceptables en termes de rendement, de taux d'échec, de coût, de latence et de charge du réseau. Dans le cas de l'assistance à la mobilité, le taux de tâches manquées ne dépasse pas un millième de pourcent, et il est prouvé qu'il est inférieur de 88 % aux solutions de pointe en termes de durée de vie et de 92 % en termes de consommation d'énergie. Notre recherche comprend également une étude de cas de simulation réaliste qui utilise l'aménagement d'un bâtiment hospitalier intérieur à Chicago, et qui a démontré des performances acceptables. Pour authentifier et sécuriser la communication entre les dispositifs IoT et les passerelles, la thèse propose également un schéma d'authentification mobile basé sur la DTLS (Datagram Transport Layer Security) pour l'architecture IoT. Il garantit un transfert mutuel authentifié entre les dispositifs IdO mobiles et les passerelles visitées tout en évitant des poignées de main supplémentaires. La performance du système proposé est évaluée en termes de temps de transfert, de temps de traitement, de consommation d'énergie et de mémoire. Les résultats démontrent sa faisabilité pour des dispositifs à ressources limitées.

Contents

List of Publications

International Journals

- RM Abdelmoneem, A Benslimane,E Shaaban, “Mobility-Aware Task Scheduling in Cloud-Fog IoT-Based Healthcare Architectures”, Elsevier Computer Networks Journal, 107348.

International Conferences

- RM Abdelmoneem, E Shaaban, A Benslimane, “A Survey on Multi-Sensor Fusion Techniques in IoT for Healthcare”,13th IEEE International Conference on Computer Engineering and Systems (ICCES 2018), pp. 157-162.
- RM Abdelmoneem, A Benslimane, E Shaaban, S. Abdelhamid, S. Ghoneim, “A cloud-fog based architecture for IoT applications dedicated to healthcare”, 2019 IEEE International Conference on Communications (ICC 2019), pp. 1-6.
- RM Abdelmoneem, A Benslimane,E Shaaban, “Mobility-Enabled Authentication Scheme for IoT architecture”, submitted to IEEE ICC 2021.

List of Figures

*

List of Tables

*

Chapter 1

Introduction

1.1 Internet of Things

Internet of things (IoT) is defined as the communication between multiple heterogeneous devices, through different communication technologies, and to the Internet, for the purpose of proposing variety of services to be applied in a variety of applications. IoT is a multi-disciplinary field that utilizes the Internet in addition to different fields of computer sciences all together to achieve the required benefits of system intelligence and connectivity. Networking, robotics, embedded systems, machine learning, computer interfacing, and security are examples of computer sciences that are used to allow IoT devices to collect, exchange, process, and share information and transfer it to a valuable meaning without human intervention. The IoT devices that can be deployed depend on the application, but can mainly be one of smart home appliances, environmental and wearable body sensors, intelligent medical objects, and electronic devices such as smart meters. Internet of things have been utilized and exploited in multiple applications ranging from small scaled home environment to larger city's implementation. Among these applications, the most popular are smart home, smart city, transportation, and healthcare. However, a growing interest and focus on Internet of things for healthcare is adopted by the healthcare sector. Multiple factors are affecting this orientation such as the rapid investment in economical smart medical devices and high-speed technologies in addition to the research rising in advanced IoT infrastructures focusing on active patient-centric care. This in turn leads to the evolution of term Internet of medical things (IoMT) which potentiate the leverage of IoT dedicated to healthcare

and medical technologies.

1.2 Problem Statement

Healthcare data has stringent requirements for quality of service (QoS), security, confidentiality, availability to authorized users, traceability of access, and long-term preservation. All these requirements are accompanied with an expected growth in the number of IoT devices. According to IDC (International Data Corporation), IoT market will grow to 75.4 billion IoT connected devices in 2025, generating 79.4 zettabytes (ZB) of data, where healthcare is forming the highest percentage of about 41% of this market surpassing other segments such as manufacturing, vehicles, agriculture, etc. [4], [5]. This prediction poses the challenge of handling this big volume of generated data in terms of storage and processing in accordance with the diversified nature of this data . Adding to that, the challenge of medical emergency cases that are sensitive to latency. Consequently, this brings up the need for the design and implementation of architectures that support these challenges.

1.3 Contributions of Thesis

This research aims at proposing an inter-operable cloud-fog based IoT architecture for healthcare. The proposed architecture supports the mobility of the patients as well as the diversity of the medical cases. Taking into consideration the constraints of the application and the intended outcomes, our contribution can be summarized as follows:

1. Present the proposed architecture, environmental context, and user context, the features of the individual modules, and the interconnection between the different underlying modules. The networking technology between layers is defined and clarified.

2. Propose, design and implement a task scheduling and allocation approach to effectively balance healthcare tasks distribution with the objectives of minimizing the latency, and ensuring reliability such that critical tasks can meet their deadlines. We compare and evaluate different reallocation methodologies and present the results.

3. Analyze different mobility-support and hand-off mechanisms, and hence propose and implement a technique that handles the mobility. In this way, sensing coverage and connectivity in the proposed architecture can be enhanced even and while the patient is moving without threatening the health of the patient. The proposed approach is two-fold: (a) the first is to guarantee the connectivity of the mobile patients to their gateways through proposing and implementing an RSS based handoff mechanism. (b) the second is to guarantee the continuous processing of healthcare services specially eliminating the latency problems associated with the change in the location and timing of the data aggregated and stored at the fog devices. This point is accomplished by proposing a mobility-aware task scheduling and allocation approach (MobMBAR). In the context of this part, we compare against the different existing approaches for IoT based Healthcare such as cloud computing only, cloud-fog architecture, in addition to other scheduling technique. We also propose a case study to evaluate (MobMBAR) in real conditions.

4. Propose and implement a scheme to maintain privacy while handling the mobility of the patients and their attached IoT devices. We design, implement, and evaluate a mobility-enabled datagram transport layer security (DTLS)-based authentication scheme for dynamic mobile IoT environment.

1.4 Thesis Outline

The thesis is organized as follows:

- Chapter 1 provides an overview about Internet of things for healthcare, problem statement, motivation and contribution of our presented work.

- Chapter 2 exposes the previous and current explorations of IoT for healthcare including applications, sensors, architectures, and network technologies through presenting the literature investigating these domains.

- Chapter 3 reviews task scheduling and allocation in IoT through describing the requirements, optimization metrics, and recent researches.

- Chapter 4 reviews security and privacy in IoT healthcare. It lists the requirements and surveys existing protocols and mechanisms.
- Chapter 5 presents the proposed inter-operable cloud-fog based IoT architecture for healthcare and scheduling and allocation approach, implementation details, and performance evaluation.
- Chapter 6 presents the extension of the architecture to support the mobility of the patient, performance evaluation, and a comparison against different architectures and approaches in the vicinity of integrated Cloud-Fog IoT and Cloud-Only IoT. Furthermore, it provides a case study simulation in real conditions.
- Chapter 7 presents a privacy preserving approach within the scope of the proposed architecture and its evaluation.
- Chapter 8 concludes the work and recommends the future work.

Chapter 2

Literature Review

At present, most medical technical devices come with some form of connectivity, from wearable devices such as biosensors to X-ray machines with Wi-Fi or Bluetooth. This allowed IoT to be the key enabler in healthcare industry through the integration of medical devices with IoT. IoT dedicated for healthcare aims at improving the quality of life for elder and unwell people by facilitating their movement, helping them in their daily activity routines, besides supporting them in their healthcare treatment plan in hospitals and homes. The main advantages brought to healthcare domain by IoT are the reduction of costs and time to perform healthcare processes and tasks, and the provision of fine-grained healthcare activities and decisions coping with the rising number of medical cases while preserving the number of care givers. There is a growing body of literature that recognizes how IoT promotes healthcare. This chapter proposes different aspects of this wide adoption.

2.1 IoT Healthcare Applications

IoT healthcare applications emerge specific QoS requirements that differentiate them from other IoT applications. Delay sensitivity, time criticality, and user mobility features are of the crucial features. In addition, data collection and processing are prone to real-time variations due to multiple reasons. First of all, the data collected from body sensors, which are used as an indication for the vital signs of the patient, has different levels of priorities. Because medically-wise, some vital signs indicate emergent status of the body. These data may need to be collected and/or processed in varying order. Secondly, these priorities change dynamically based on the medical status of the patient. For example, if the

processed vital signs indicates an emergent case for the patient, the rate of the collection may increase or new tasks may be generated to be processed immediately such as controlling a specific actuator or sending an alarm message. In addition, if the infrastructure comprises of multiple patients with different medical cases, the dynamic nature and the variation have to be handled more appropriately.

The key characteristics of IoT healthcare application include:

- Heterogeneity of devices: IoT healthcare applications demand the usage of wide range of heterogeneous sensors. The first type is the body sensors (wearable or implantable) classified into biomedical and activity sensors. Biomedical sensors are mainly targeted to record the health and physiological vital signs of the patient. Activity sensors are any body worn devices that can help for determining the location, posture and position of the patient. Examples on body sensors are heart rate, blood pressure, Electrocardiography (ECG), Electromyography (EMG), gyroscope, etc. The second type is the environmental sensors for measuring the contextual environment around the patient. They can be mechanical, acoustic, optical, chemical, force, proximity and even imaging sensors. Examples are temperature, humidity, infrared, contact, cameras, etc. A list of possible sensors that are utilized in IoT healthcare and their measurements are shown in table 2.1. The Heterogeneity can also exists in the connectivity capabilities whether bluetooth, zigbee or wifi, etc.

Table 2.1: IoT healthcare sensors

Sensor	**Measurement**	**Unit**
Wearable - Biomedical		
temperature	measures body or skin temperature	Celsius degree
ElectroCardioGgram (ECG) electrode	records electrical activity of heart (to infer heart beat)	Volts per second
ElectroEncephaloGram (EEG) electrode	records electrical activity of brain	Volts per second

ElectroMyoGram (EMG) electrode	records electrical activity through muscles	Volts per second
pulse oximeter	measures the oxygen saturation in the blood of the patient	percentage
glucose level	calculates the amount of sugar in the blood	electrical signal (Volt)
respiration piezoelectric	monitors abdominal or thoracic breathing	Breaths Per Minute (BPM)
oscillometric device	provides automated oscillometric blood pressure measurement	millimeters of mercury (mmHg)
Wearable - Activity		
accelerometer	measure the linear acceleration	meter per second^2
gyroscope	measures angular velocity to infer orientation	Degrees per second
magnetometer (magnetic sensor)	measures the magnetic field for physical axes(x, y, z) to infer orientation	micro Tesla
Environmental		
ambient temperature	measures the degree of	Celsius degree

	indoor temperature	
relative humidity	measure the percentage of indoor humidity	percentage
light	measure the indoor illumination	Luminous flUX per unit area (LUX)
electrochemical carbon dioxide	measure the quantity of CO2 present in the air	parts per million
gas sensor	detects the presence or concentration of gases in the atmosphere	parts per million or (percentage)
imager (image sensor)	captures the optical information	digital or analogue signals
microphone	measures the intensity of the sound in the air	decible (dB)
pressure	detects physical pressure on objects	Newtons
smart tiles	detects physical pressure on floor	Newtons
Passive InfraRred (PIR)	measures infrared light radiating from objects to infer motion	Volt with high value represents motion

- Latency sensitivity: Healthcare implementations usually implies taking care of the health of

the person on the short term or the long term, In both cases, it is considered an emergency to prevent the patients from falling into a health risk specially life threatening. To support medical emergency, it is required to provide accurate monitoring, quick detection of risk situations, and real-time care response from system and/or caregiver. The delay in any of these phases exposes the patient and the system to very high danger.

- Big volume of data: The raw data is collected from numerous sensors existing in the environment. In addition, the data is collected with high sampling rate over a long duration that can reach up to days. This production of massive amount of data requires the system to analyze, store, and to infer notable outcomes. The processing of big data is not an easy task that requires acute quick task scheduling, allocation, and distribution. Similarly, this voluminous data demand storage and retrieval in addition to the management of such operations whether in the cloud or in the servers at the site (edge), or the integration of both proposed as a hybrid methodology.

- High security and privacy requirement: One of the big challenges in IoT healthcare is maintaining the security of the data gathered and the privacy of the patient served. Multiple factors have to be guaranteed such as authenticating the communicating peers and devices for the data exchanged between them to be trustworthy, and hence to protect the data. Furthermore, proper access control mechanism have to be setup and implemented to allow the client and health care providers to interact securely. Adding to that, the need to preserve the identity of the patients if required due to working on personal information. All of these parameters determines the acceptance or the abandonment of the patients to enable these systems to enter their life. Also, It provokes the legal entities towards the legislation of such projects and hence, embedding them into the society.

- Mobility: Healthcare applications seek to fulfill the human nature of the beneficiary patients who is at the end a person unwilling to be mobility restricted. This is referred to as the quality of life. In other terms, the quality of service proposed by the IoT healthcare application represented in have to be accompanied by the quality of life of the patient. This in turn mean the guarantee of continuous supply of the networking, processing reliable health services whether monitoring

or treatment. Hence, supporting the mobility characteristic in an IoT healthcare application is of great demand.

2.1.1 Remote Monitoring and Abnormality Detection

Remote monitoring is defined as long-distance clinical healthcare applications that observe vital signs of the patient such as blood pressure, glucose level, oxygen saturation, heart rate, ECG (electrocardiogram), EMG (electromyogram), and EMG (electroencephalogram) signs in addition to activity. Abnormality or Anomaly detection refers to the recognition of unpredicted health state or activity of the patient. An example can be a measurement of a vital measurement passing a threshold such as the heart rate. The detection of such cases is considered a second level of monitoring which helps in proposing immediate action in critical situations.

A patient activity monitoring project entitled SPHERE (Sensor Platform for Healthcare in Residential Environment) highlights an e-health platform based on a collaborative IoT research by three universities in UK [6]. The project encompasses three types of sensors: wearable, vision and environmental for the purpose of monitoring the movement of the patient in addition to the surrounding environment. The environmental sensor BLE (Bluetooth low energy), IEEE 802.15.4 equipped board is responsible of sensing the temperature, humidity, light, air pressure, motion, and noise level sensing. The wearable unit is an accelerometer sensor based board responsible of tracking the activity. Finally, the RGB-D camera unit is considered the vision board. Whilst in [7], the authors propose a mobile health application for monitoring glucose level in the patients suffering diabetes. They exploit IEEE80.15.4 compliant wireless device equipped with glucose sensors in a constrained environment. An energy saving healthcare IoT approach is proposed in [8]. Based on RFID system, their work is to be able to monitor vital signs such as glucose level, heart beat, and blood pressure. The authors in [9] propose a monitoring system for breathing-problems patients. The clinical platform is composed of gateway and mobile vital sign monitoring board capable of sensing patient's oxygen saturation.

ECG monitoring is a common application which is addressed in multiple works such as [10], [11], [12], [13] and [14]. These application tends to propose and use feature extraction algorithms and classifiers to detect notable ECG information such as QRS, QT and PR intervals. Fast detection of

heart attack symptoms helps in preventing the deterioration in patient's health condition. The work in these article vary between IoT applications that depend only on the ECG signal inferred from the wearable sensors, and exploiting the aid of the surrounding environmental signal to strengthens the decision taken about the patient.

An IoT health project named TIHM(technology integrated health management) is proposed in [15] for health monitoring. It helps to monitor dementia patients - usually elder people - who need a continuous care due to their memorizing, thinking and socializing disabilities. They can fall into social and health problems in addition to probability of being alone which in turn requires automatic health support. TIHM utilizes wearable sensors: blood pressure, pulse oximetry, body temperature, weight, body water, in addition to environmental sensors: PIR, door sensor, room temperature, room humidity, and object occupancy to infer the status of the patient by applying rule-based reasoning algorithms. Similar work is proposed in [16] which describes a general comprehensive IoT architecture for monitoring the patients. The main focus of this article is on the layers and components of the architecture model instead of the medical case. This will be discussed in more details in section 2.2.3. In addition, healthcare monitoring applications occasionally include treatment followup plans such as insulin delivery and cancer treatment.

2.1.2 Disease Diagnosis and Classification

Healthcare diagnosis aims at disclosure of specific disease or health situation related to the patient through utilizing IoT system. This IoT diagnostic system gathers the health information from the wearable or implantable sensors and then utilize the medically developed methods to predict a disease based on the council of the medical expert.

The authors in [17] propose a work to predict the diabetes patients in addition to classifying the severity and level of the disease. They propose a Neural Classifier based on fuzzy rule where they apply the algorithm on a data set from UCI Machine Learning Repository stored at the cloud that was collected from an IoT wearable sensors. In [18], the authors tend to diagnose the patient's disease based on vital data collected from wearable and implantable sensors such as ECG,EEG, and blood pressure. They firstly generate a profile for the user containing general and health information such

as the age, gender, weight, systolic blood pressure, etc. Next, they propose an algorithm to analyze the potential probability of the diseases with its level. They build their knowledge based on medical experiences in books and from advisors. Another work is presented in [19] that targets to diagnose heart disease automatically based on an integration between IoT and fog computing. They utilize deep learning with that is adapted to fit in the constrained infrastructure of IoT to perform heart analysis for the patient. The multimodality sensing boards collect the vital health data: ECG,EEG, EMG, temperature, respiration rate, glucose level, and oxygen level in addition to activity and environmental data.

2.1.3 Activity and Fall Detection

Similar to the recognition of the abnormal vital signs of the patient, it is important to recognize the abnormal movements or activities of the person to be able to provide protection or propose quick helping response. Several articles proposed methods to record and infer the activities of the patient specially the falling.

The authors in [20] aims at recognizing the critical changes in bipolar disorder patients which can also be called manic depression where patients suffer sharp swings in their modes. In its negativity round, this illness can pave to a great risk of bad results on the patient's behavior. The work tends to record social, physical and travel data of the patient's day such as phone call's number, duration, sound, speech, and voice data captured from patient's smartphone. Next, machine learning algorithms such as linear discriminant analysis and naïve Bayes classifier are applied to extract the features. In [21], a system is proposed to detect the fall of the patient depending upon wearable sensors in and IoT environment in an energy efficient way. Through deploying motion sensors: accelerometer, magnetometer, and gyroscope, the authors could infer rate of velocity change, absolute magnetic intensity, and orientation (angular motion) of the body respectively. These data are combined to calculate falling features parameters through vector operations such as sum vector magnitude (SVM) and differential SVM which in turn are used to detect the fall. The authors study different factors affecting the energy consumption of the sensor device such as: sampling rate, communication system, and divergent conditions of transmission. Prime survey are conducted focusing on fall detection and fall prevention as a

standalone domain that covers the methodologies, architectures and sensors in IoT. Examples on such survey can be found in [22], [23] and [24].

2.1.4 Sensor Fusion

Multi-sensor fusion techniques are the techniques that combine several unrelated devices and sources of data to be processed together for better quality of services. Multi-sensor data fusion is presented to obtain better results from the aforementioned different sources [25], [26], [27], [28], [29]. These multi-sensor fusion approaches increase the reliability and robustness of the healtchcare systems by reducing the threatens posed by various malfunctions in the sensors and environment itself such as the power and communication [30]. In addition, global decisions can be taken based on the dependability between the data sources.

Several surveys have been conducted on multi-sensor fusion techniques. A survey on multi-sensor fusion techniques in body sensor networks can be found in [31]. The authors categorize the fusion process according to the level where the fusion is performed: data-level, feature-level or decision-level. The authors focus on fusion of the sensors mounted on the body for the purposes of different applications including the recognition of activities and emotions, and general health. But how the multi-sensor fusion process affects the different healthcare applications is not tackled such as studying the influence of numerous techniques on the detection and diagnosis of diseases. Another survey paper in [32] discusses the data fusion mathematical methods. The authors focus on the probabilistic, artificial intelligence and evidence based methods for data fusion in applications including multi-target tracking, environmental monitoring and remote sensing without reviewing any techniques applied on healthcare applications. The authors discussed environmental challenges in IoT such as the distribution of the environment, heterogeneity and non-linearity and its affection on the process of data fusion. In [33], the authors propose a survey on data fusion in IoT in correspondence with the term context awareness. The authors categorizes the data fusion technique to three levels as reviewed in [31]. They propose explicit details of the advantages and limitations without referring to the application where the fusion technique is implemented. The paper in [34] proposes the state of the art of the data fusion techniques applied to the sensors settled in mobile devices. These techniques are distinguished

to be suitable for the different constraints on the mobile devices such as the limited processing and power capabilities and also distinguished for working on specific sensors equipped in the mobile devices. The work in [35] presents ten parameters to evaluate sensor data fusion frameworks. Next, the authors focus on one of the proposed parameters "fusion complexity" to evaluate different proposed approaches for sensor fusion. According to authors of [30], sensor fusion can be categorized into data-level, feature-level and decision-level. In data-fusion, the raw data coming from sensors are combined directly. In feature-level fusion, the features of the data collected from the different sensors are firstly extracted and then fusion technique are applied on the features for the purpose of combining them. On the other hand, decision-level fusion includes applying data mining, machine learning and computer vision techniques on the different decisions gathered from the processing of the individual sensors for the purpose of reaching a global decision. Several techniques of decision-level sensor fusion category are used in healthcare systems for multi-modal sensors including intelligent fusion based models, e.g. fuzzy logic based techniques, probabilistic and statistical based models e.g. Bayesian reasoning and Markov Decision Process and Dempster Shafer theory. Other models such as data mining based models and threshold technique based models also exist. The following paragraph focuses on the methodologies of decision-level sensor fusion category.

The main metrics authors used for measuring the performance of the decision-level fusion techniques are: sensitivity, specificity, error rate and perfect classification. Sensitivity measures the proportion of positives that are correctly identified. In the healthcare domain, sensitivity means the ability of a test to correctly identify the status of the patient such as correctly reporting a disease, health condition, or falling in case of fall detection applications, etc. Whereas specificity measures the proportion of negatives that are correctly identified. Respectively in healthcare domain, it is the ability of the test to correctly identify those without the disease or not having specific health condition or not falling in case of fall detection applications. Perfect classification is one of the parameters that are used to validate a classifier and is defined as the number of correctly classified samples to the total number of samples [30]. While error rate is the number of wrongly classified samples to the total number of samples. A summary of fusion techniques in IoT healthcare is presented in table 2.2. The choice of the sensors differs according to the type of the disease and the type of application. Most

Table 2.2: A summary of fusion techniques in IoT healthcare

Ref	Technique	Application	sensors used	Sensitivity	Specificity	Implementation
[30]	Fuzzy logic	Healthcare monitoring of patient	Set of microphones, Wearable, Infrared sensors, Monitoring sensors	97%	NR	Both
[36]	Fuzzy logic	Classifying mental state of patient	Wearable {heart rate, respiration rate, oxygen saturation, finger temperature, carbon dioxide}	75%	100%	Real platform
[37]	Fuzzy logic	Detection of asthma attack	Wearable{heart rate, respiration rate, oxygen saturation}	NR	NR	Simulator
[38]	Fuzzy logic	Monitoring patient's health	Wearable{heart rate, respiration rate, temperature, systolic blood pressure}	NR	NR	Simulator
[39]	Baysian	Fall detection	Wearable accelerometer	87.5%	100%	Real platform
[40]	Baysian	Remote monitoring & fall detection	Temperature, Wearable{electrocardiography, accelerometer}	WEKA: 87.5% Real: 100%	NR	Both
[26]	Baysian	Location prediction Activity recognition	Passive infrared, Wearable accelerometer & Camera video	85%	94%	Real platform
[41]	Baysian	Fall detection	Wearable {accelerometer, heart rate, SpO2}, Environmental {PIR motion, door contact, pressure mats, power usage detectors}	90%	100%	Simulator
[42]	HMM	Detection of physiological abnormalities	Wearable vital-signs sensors, Environmental binary sensors	91.4%	NR	Real platform
[27]	HMM	Detecting possible dementia	Passive Infra-red, Wearable motion sensors	log-likelihood:94% Kullback-Leibler:100%	NR	Simulator
[28]	Dempster Shafer	Fall detection	Vital signs: RFPAT[43], Environmental: GARDIAN[44]	94%	100%	Real platform
[45]	Threshold technique	Fall detection	Wearable accelerometer, Environmental sensors { temperature, infrared motion, pressure, magnetic }	100%	NR%	Real platform
[46]	Threshold technique	Monitoring Alzheimer patients and fall detection	Motion sensors, Wearable patch, Anchor points	87.5%	98.7%	Real platform

of applications that perform health monitoring uses all the vital signs sensors, while applications that perform fall detection mainly depend on the posture sensors such as the tri-axial wearable accelerometer. On the other side, activity monitoring applications rely on environmental sensors such as optical and mechanical. The processing is mainly centralized in almost all of the proposed techniques where the data from the sensors are sent to the processing unit. The performance metrics shown in table 2.2 are guiding factors to compare the quality of different decision-level techniques. Table 2.3 lists the pros and cons of different decision-level fusion techniques in healthcare domain.

2.2 IoT Architecture for Healthcare Systems

2.2.1 Cloud Computing Vs fog Computing

IoT architectures for healthcare are extending from traditional architectures to more comprehensive ones to be able to help in the medical sector and provide better services to the healthcare domain. IoT architectures for healthcare describe a context of a hospitalized environment, assisted living environment such as convalescent home or even a personal home that comprises multiple layers and devices where IoT devices settle in the lowest layer. IoT devices include multiple medical body sensors and actuators, motion devices, and environmental sensors. These devices are connected to communication devices in upper layer(s) which are responsible for transmitting the health data from their source of generation until reaching their permanent storage at the destination layer (usually cloud storage). This data is further queried or deeply analyzed by the doctors and the care givers. The nature of the data stored and analysed at the cloud can be classified into raw data and processed data. Data is raw if the design of architecture is based on cloud computing only. On the contrast, the data is fully or partially processed in the case that the architecture embraces other computing infrastructure such as fog computing.

In cloud computing explorations, datacenters at the cloud side are responsible of performing the computing and storing the data. This in turn requires continuous sending of large sized data and demands between the cloud and the IoT devices over a far distance through the Internet. Cloud based architectures were considered the base to current IoT enabled solutions till recently where attempts are

Table 2.3: Comparison of decision-level fusion techniques in healthcare

Technique	**Pros**	**Cons**
Fuzzy logic	• Deals with ambiguous input sensors • Flexible • Adaptable • Can be merged with other techniques (hybrid techniques) • Most used in monitoring and classification applications	• Mismanage the dependencies between the input senosrs
Bayesian	• Simple • Dependencies between sensor inputs are allowed • Behaves better in multi-sensor inputs • Most used in fall detection applications	• Limited scalability • Requires prior knowledge of medical information
Markov Process	• Usually built based on binary sensor inputs (eliminating ambiguity, easy environment setup) • Most used in detection and predication of daily activities and abnormality detection	• High computational complexity • Usually combined with other fusion method for inferring the final decision
Dempster-Shafer based	• Deals with uncertainties • Works well with limited number of sensor inputs despite being heterogeneous	• Computational overhead increases proportional with number of sensors
Threshold technique	• Good results when applied to specific case • Can be applied in different applications	• Case based scenarios • Limited papers addressing healthcare domain

made so that cloud computations are pushed to network boundaries for which the term fog computing is established.

Due to the advances in Information and Communication Technology (ICT) devices in terms of computation, communication and storage, the fog computing term is introduced where the computation is transformed to the edge of the network. Fog computing refers to a group of small scale data centers installed as a middle of the road layer between the cloud and the things (IoT devices) [47]. These fog devices are traditional devices such as wired or wireless switches and routers, in addition to microcomputers. Fog computing is used in several domains such as in [48], where fog devices are utilized to support peer-to-peer cashing in content delivery networks (CDN). These fog devices have computation, storage and networking capabilities. Fog devices have multiple functions to perform such as handling the computation of raw data produced by sensors, communicating with similar corresponding fog devices in the same layer or other layers in the hierarchy for further or combined processing. Examples of local processes that can be performed on fog devices are described in [49]. In the environment of fog computing, the intelligence and processing are placed at the local area network (LAN).

The architecture is popularized as integrated Cloud-Fog based architecture in the case that the IoT architecture is jointly fog-computing and cloud-computing based. Cloud-Fog integrated IoT architectures are recently proposed targeting many applications including healthcare. Healthcare services with high quality are provided through the collaboration of heterogeneous computational devices existing in the fog environment with the processing devices in the cloud [50]. High-grade services of healthcare therefore can be accomplished through on-site execution of the emergent and time sensitive tasks. The motivation for building such kind of paradigm is achieving the trade-off between larger computing and storage resources available on the cloud, and the reduced communication latency, bandwidth and monetary costs in addition to security in fog environment. Table 2.4 describes the QoS requirements and their relation with fog and cloud computing characteristics for fulfilling the demands of IoT healthcare application.

Table 2.4: QoS Requirements and their relation with fog and cloud computing characteristics

QoS Requirements	**Fog Computing Characteristics**	**Cloud Computing Characteristics**
Scalability	Distributed deployment support	Centralized deployment support
Delay sensitivity	Low latency due to on-site processing	Delay tolerant
Energy efficiency	Reduced energy consumption due to short range communication	Increased energy consumption due to long range communication
Reliability	Small and medium sized tasks	Computing intensive tasks
Geographical arrangement	Distributed	Centralized
Privacy	Environment controlled to some level due to local connectivity	Data needs higher protection

Cloud Computing

Several IoT cloud-based architectures for healthcare have been proposed. Traditional infrastructures involve a set of cloud resources accessed remotely by the users. The authors in [51] propose a smart healthcare system called Health-CPS which is intended for aiding in healthcare application services such as statistical monitoring, knowledge and prediction. The architecture consists of three layers: 1) the unified data collection layer where individual activity and emotion data can be collected from the IoT environment in addition to other clinical and research data, 2) the data management layer on the cloud where data from multiple heterogeneous sources is analyzed and stored, 3) the application service layer where APIs are provided for the users to access the system and retrieve information. In [52], a system is introduced to monitor patients with chronic diseases using a mobile device and a server. The architecture is divided into two components. The client component is a mobile device for interfacing with the patient. The server component contains context detector and a reasoning engine responsible for performing the inferences on the data. In [53], the authors propose a remote monitoring and processing system based on bio-potential signals from the patient. The architecture is composed of IoT body sensing devices, the gateway and the cloud. The data is transmitted to the cloud where a processing classification algorithm is performed. The user can access the results of the analysis through a cloud-based application.

The work in [54] proposes the Ubiquitous Healthcare System (UHS) architecture intended for the elderly. The proposed system works on several factors aiming to help the elderly in terms of providing home care services, emergency assistance, and remote medical services. Two systems are proposed which are the Web-based User Remote Management Service (WURMS) and the Multimodal Interactive Computation Services (MICS). The architecture is composed of the IoT sensor layer, (push buttons,cameras, sound sensor, physiological sensors) connected to the cloud servers layer where the processing is performed. The service provided requires the implementation of audio and visual techniques on the cloud including speech/sound recognition, speaker identification, face identification, sound source estimation, text to speech (TTS), and event recognition. In [55], a cloud-based healthcare system is built to monitor the status of the patient through the measurement of ECG signals. A feedback medication modification is returned to the patient through a mobile device. The architecture

is composed of three parts: a data acquisition module, a mobile system, and a cloud computing web server module.

In [56], a cloud-based preventive health monitoring system is proposed to help users maintain and prevent their health from catching a disease. Cloud services contain the central database holding patient's health status and accessed from multiple different doctors/experts through web services. The health status of the users such as blood pressure, body fat, weight and agility are analyzed and monitored by health assessment systems on the cloud. The assessment systems perform evaluation of the health risk factors. If exist, the user can work out to maintain their health. Traditional IoT architectures that are based on cloud computing adds several limitations to the current healthcare systems despite their huge capabilities. Limitations include huge processing cost due to reserving the cloud processing devices and data transmission costs and high overhead degrading network performance. Moreover, extra time connecting to the cloud adds a high latency limitation which is unacceptable for the criticality and time sensitivity nature of the healthcare applications.

Fog Computing

On the other hand, some authors proposed fog computing based healthcare architectures. A fog based system for e-health is proposed in [57]. The system aims for detecting abnormal events such as patient falls and gas leaking in a home environment. In the proposed architecture, the fog devices stores the medical data gathered from the home devices and sensors in addition to performing the signal processing. The cloud devices receives and stores the meta data sent from the fog devices. Two algorithms are implemented for fall detection and gas detection. This system is practically designed and implemented only for two static scenarios. In [58], authors propose using fog computing devices (represented by mobile device held by the patient) along with cloud computing servers for a pervasive fall detection health system. The authors propose a distributed fall detection algorithm where part 1 of the algorithm which is performed on the mobile device processes light weight computation for threshold based fall detection algorithm. Part2 performs analytics based fall detection algorithm which has higher accuracy but is complex to be implemented on smart phones. The final decision that fires the alarm at the patient side is based on results obtained from both implementations specifically the

edge (mobile) based when connection to the cloud is disconnected or have problems.

Despite that the aforementioned two fog-cloud based papers are only limited to simple processing that can be processed on the mobile devices, other proposals define the fog devices and their processes as being more complex such as [59][60][61]. In [59], the authors propose a fog computing architecture for the aim of clinical speech processing for patients with Parkinson's disease. Processing is performed on a fog computer where the cloud is only used for the long term storage. The data are gathered though a smart watch and then the processing is performed on a fog computer (Intel Edision) in the home. The cloud layer is used for the storage of the extracted features for longterm analysis. Another work is proposed in [60] for Chronic Obstructive Pulmonary Disease patients. The authors focus on the idea of the existence of real-time processing provided by the fog to cloud (F2C) concept. The work lacks the details about the processing devices, place, capabilities, and how tasks are handled between the fog and cloud. Generally speaking, the existing work is very specific to certain medical use cases. In addition, all complex processing is performed on the cloud layer even if it is emergent and greatly affects the latency.

2.2.2 Cloud-Fog IoT Architectures for Healthcare

Recent research directions in IoT architectures are moving towards exploiting fog resources along with cloud resources for the purpose of better QoS [62], [63], [64]. In [14], an approach is proposed for health monitoring relying on the fog layer to save bandwidth of the network. In their work, the fog layer performs event classification based on Bayesian Belief Network (BBN) classification method. The classifier defines the status of the patient into normal or emergent in which a DOI (degree of impact) output is produced. If DOI is over specific threshold, all the signals are sent to the cloud for processing. Hence, all the health services are processed on the cloud layer, and fog computing is utilized for determining whether or not tasks are going to be executed on the cloud. The main sensors they are body temperature, blood pressure, heart rate and glucose level. The authors measure the accuracy of the classification method used and the classification time.

The authors in [61] propose an architecture where a one low powered fog computer performs data onsite processing. In this work, the authors utilize the fog gateway computer to perform one of

defined tasks: dynamic time wrapping, clinical speech processing chain, compression, and data reduction. Only one service is performed on that fog device depending on the medical case chosen for the case study. Utilizing this partial processing on the fog computer, the authors can reduce the size of data permanently sent the cloud. Hence, their QoS can be achieved through bandwidth utilization and energy efficiency by the means of reduction of transmission power consumed when sending to cloud layer. In [59], the authors perform analysis of speech tele-treatment in the fog environment through presenting fog computing interface (FIT). The authors utilize Intel Edison tablet representing a fog device. The fog device collects data from one patient and performs speech processing (feature extraction) and then the results are forwarded to the cloud layer. The purpose is enabling the cloud layer to work on the extracted features directly. The authors evaluate their performance through the measurement of speech features such as loudness and spectral centroid, and the impact of the amplitude of the speech signal on these factors.

The approach used in [58], where the fog device is represented by a smartphone, is similar to that used by [59] in partitioning the analysis processes between the fog device and the cloud. Based on prefixed installation, the fog smart phone is responsible for executing a threshold based fall detection algorithm. Whereas a more accurate and complex fall detection algorithm based on non-linear time series analysis is used on the cloud layer. Feedback messages is mutually exchanged between both devices in order to produce the best accuracy. The authors evaluate the accuracy of system through measuring the sensitivity and specificity of the classification algorithms performed. In addition, the energy consumed by smartphone during the sensing (gyroscope and accelerometer sensors) and computation is measured. Finally, the authors also measure the response time which indicates the time for the detection of the activity of the patient. The work in [57] proposes an architecture for smart home application specifically fall and gas leaking detection. It uses a standardized method that repeatedly executes thresholding algorithms implemented on the fog device. Thus, their work considers local processing on fog device (Arduino Uno board) while the cloud layer is used only for storing patient medical information. The QoS that the authors take into consideration is the accuracy of detection which corresponds to percentage of correctly classified falls for fall detection, and correctly classifying the gas leakage into 1 of 10 classes. In the same way, other example in [60] introduces assistant

systems for Chronic Obstructive Pulmonary Disease (COPD). The authors refers that the health tasks are executed exploiting fog computing without describing an architecture for linking between the patient, IoT device, fog device, or cloud device. The tasks are mainly concentrated on measuring the oxygen saturation, the patient's activity, and controlling the oxygen doses provided to the patient. Their target objective is medical wise where they study how the mobility of the patient affects the oxygen measurement and doses values which unrelated to any networking or computation QoSs.

To conclude, Cloud-Fog IoT architectures dedicated for healthcare published in many recent research are addressing specific medical cases. Moreover, the environment is statically setup and configured throughout the application period. This means the same tasks are executed on the same fog device or cloud device without taking into consideration the status and number of the patient(s), or the diversity of the nature, number and heterogeneity of task(s) at a specific time. Hence, adaptable architectures that support multiple patients or different medical cases or diseases are rare despite that these architecture are extremely needed in medical establishments such as hospitals or retirement homes. Also, there is a lack in works that perform dynamic distribution of the health tasks among the processing devices.

2.2.3 Networking Technologies in IoT for Healthcare

As discussed in the previous section, IoT architectures for healthcare comprise various communication requirements between different communicating entities. Multiple factors affect the choice of networking technology such as deployment area, financial costs, and number of patients/users. however, the most technical parameters influencing the adoption of one technology over the other are the device's capabilities and role in the healthcare application, whether IoT device, fog, or cloud, in addition to the purpose of communication. Wired technology can be utilized between non moving objects for preserving the reliability and security of the link. For example, wired communication between fog devices (Local Area Networks communication), or between fog and cloud devices (Ethernet link) can be used. On the other hand, in case of mobility, specially IoT mobile devices, or hard/costly wired configurations, wireless technologies can be adapted. Wireless technologies are classified based on the transmission range, data rate, bandwidth, and power consumption. Wireless Personal Area Networks

(WPAN) and Wireless Local Area Networks (WLAN) share proximate transmission range from tens of meters up to 100 meters, while Wireless Local Area Networks (WLAN) supplies higher data rate. On the other hand, Wireless Wide Area Networks (WWAN) covers longer transmission ranges, but varies in the data rates and hence can be categorized into two categories: cellular, and low power wide area (LPWA). low power wide area (LPWA) are networks delivering high transmissin range close to cellular networks with lower data rate. Examples on standards being developed under LPWA are LoRaWAN, LTE-M and NB-IoT (NarrowBand-Internet of Things). Here, we discuss some of the common wireless protocols utilized in these networks and adopted in IoT architectures in section .

- Wi-Fi is the wireless local area network (WLAN) IEEE 802.11 specification based technology. It is a featured by its high data rate ranging from tens of Mb/s to even some Gb/s and medium transmission rate less than 100m. It is preferred to be utilized in devices with medium capabilities due to high power consumption. However, in applications that require fast transmission of data due to latency prone and low cost infrastructures, Wi-Fi can be beneficial.

- Bluetooth low energy (BLE) is a short-range energy efficient wireless personal area (WPAN) technology designed for IoT applications. It is an enhancement over the Bluetooth wireless technology standard. Despite its low data rate represented in less than 1 Mb/s, it can be utilized on applications requiring low-power consumption where devices are settled with close proximity, and at the same time non restricted to slow data transmission speed.

- Zigbee is considered a common inexpensive protocol with larger transmission range than Bluetooth while having lower data rate represented in few hundreds of Kb/s. It is based on IEEE 802.15.4 standard industry implemented protocol. It can be utilized for devices with extremely energy limited specifications and limited data transfer applications at low transfer rates.

- Z-wave is WPAN protocol that is close in targets and specifications to Zigbee. Z-wave has shorter communication range than Zigbee that reaches a maximum of about 30 meters while proposing lower data rates. Despite that Zigbee is more commonly used and adapted in IoT, Z-wave can be featured by its simplicity in implementation.

- Cellular (3G,4G,5G)is a very popular licensed long-range wireless connectivity. It is commonly referred to as LTE(long term evolution) representing the currently used fourth generation 4G. The fifth generation 5G is predicted to reach the market as the next successor generation with faster connectivity and more capacity. cellular networks usually provide high data rates in tens of Mbps transferred over the internet and longer transmission ranges. Despite that, cellular networks requires a complex costly deployment in addition to very high power consumption.

- LoRaWAN (long range WAN) is a unlicensed Low Power WAN (LPWANs) wide-range technology reaching communication between devices in terms of kilometers. However, its data rate is very low which can be considered the lowest between all the wireless technologies in IoT of around tens of Kbps. Due to this low data rate, it can demand more transmission time for long range connectivity.

- NB-IoT (NarrowBand-Internet of Things) and LTE-M are licensed versions of the LPWANs. These two protocols are new network technologies based upon cellular technology less commonly known as LoRaWAN. Featured for being battery-life saving, they also provide higher data rates than LoRaWAN reaching up to 1 Mbps.

Figure 2.1 presents the taxonomy of networking technologies in IoT embedded with their corresponding data rates and transmission ranges.

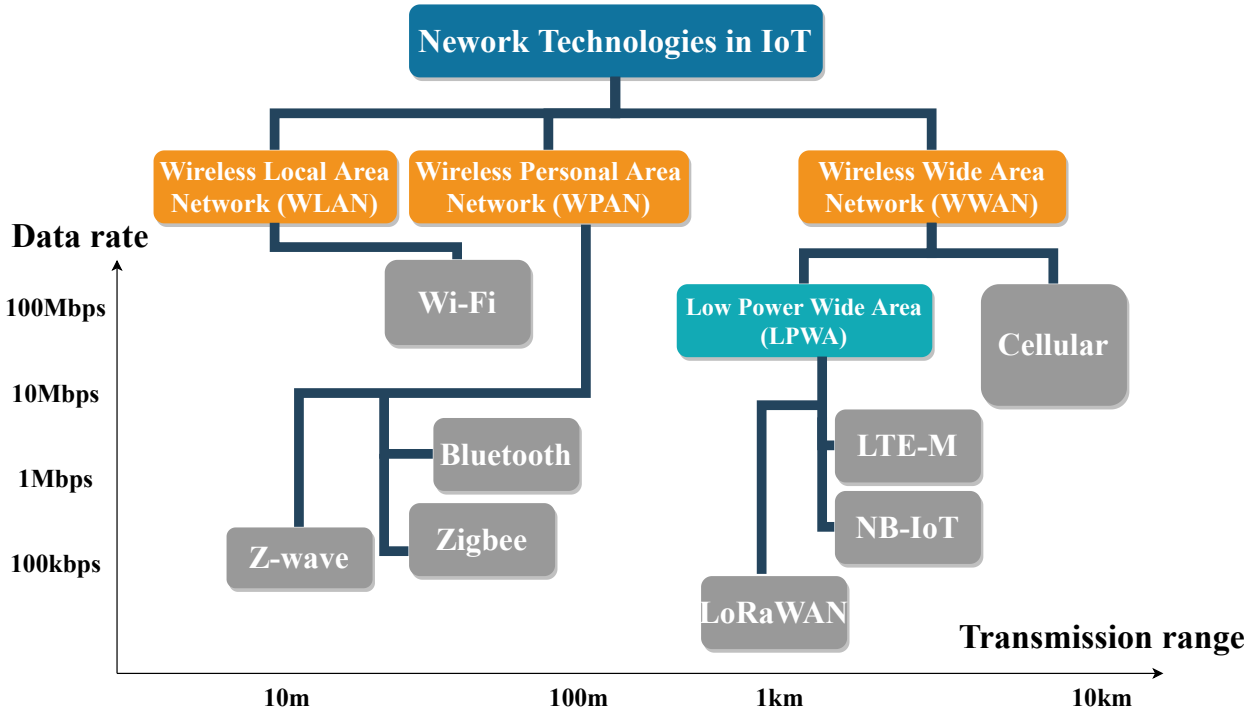


Figure 2.1: Network Technologies in IoT

Chapter 3

Task Scheduling and Allocation in IoT

Task scheduling and allocation is considered one of the most important task management processes that nominates the appropriate computing source to execute a system's task, and to order the execution of all the tasks such that the system's objective and constraint are met. Firstly, we need to define

- A task or a workload is the piece of code, job, or a service to be executed.
- Task allocation means assigning a given task to be executed and processed over a specific computing device.
- Task Scheduling is the arrangement of tasks execution over a processing device's queue in addition to defining the start and end times of the tasks according to some constraints.

Task scheduling and allocation in IoT, whether cloud computing or hybrid cloud-fog computing architectures, has to utilize the IoT distributed computing resources while taking into account the nature of the application tasks.

3.1 Requirements and Optimization Metrics

It is important that task scheduling and allocation strategy developed meets the requirements of the application where it is deployed. Here, we list some of the requirements that IoT environments poses:

- Heterogeneity: The heterogeneous nature of the sensors and IoT devices usually leads to heterogeneous tasks. The tasks can have different priorities and can variate in their computing

requirements and resources consumption. Adding to that, the computing and storage resources at the IoT environment are dynamic. This in turn poses the requirement for appropriate development of effective task management strategies e.g. dynamic scheduling and allocation of these tasks on the resources.

- Reliability: It is vital that the system is able to overcome the failure in the network and hardware without affecting the handling of the application's tasks specially the processing of the critical tasks in real-time applications such as healthcare tasks.

- Resource constraint IoT devices: One of the biggest challenges in IoT environment is the resource constraints devices whether the sensor devices or the networking and processing devices defined as fog devices. The scheduler has to take into consideration this limitation through applying workload balancing in terms of processing, energy, and storage resources. This in turn means when and how to gather the data from the constrained sensor devices, how to balance the execution of the tasks on the constrained fog devices if exists, and when to offload the tasks to the cloud without risking the reliability of the application.

- Flexibility: It can be defined as the possibility of adapting to changes in the environment such as the addition/removal of new data input sources (sensors, users, etc...), or the upgrade in one of the workloads. It is preferred that the scheduler be flexible enough to to work on changing inputs without the need for being redeveloped by the designer and the developers.

- Isolation: Isolation is a feature that must exist in the scheduling and allocation strategy in the applications where there is dependability between the issued tasks. In such situations, the conflict between the tasks sharing the same data or controlling the same actuator have to be handled by the scheduler. This can be handled by multiple methodologies such as reporting the problem to the mechanism of control access.

Another perspective to investigate is the optimization metrics needed to achieve, which is based on the application's specifications. The scheduling and allocation strategy can target one or more of the following optimization metrics. In case that the target is singular optimization metric, the scheduler is

called uni-objective, while it is defined as multi-objective scheduler if it is trying to achieve a trade-off between two or more of the goals. The following list enumerate the metric, its description, and the objective of the most common metrics measured and evaluated in IoT applications, networks and architectures:

1. Energy consumption - the energy used by the sensors, IoT devices, the computing devices, or a combination of them. It needs to be minimized.

2. Cost - monetary costs for delivering a service such as cloud resources utilization cost. It needs to be minimized.

3. Bandwidth - rate of data transfer. It is better to be maximized.

4. Throughput - the size of actual data transferred from source to destination. It needs to be maximized.

5. Latency - the time it takes the packet to be transferred from source to destination. It has to be minimized.

6. Network load /Traffic load - amount of data the whole network carry. It has to be minimized and/or balanced.

7. Network jitter- the congestion generated when packets are trying to use the same network. It needs to be minimized.

8. Fairness- defines the fair allocation of scarce resources. It has to be maximized.

9. Coverage of IoT- increasing the network coverage tends to reduce the connectivity gap in the network. It needs to be maximizes

10. Network Life- represents the time during which the network is operational. It has to be maximized.

11. Makespan - defines the time difference between the start and finish of a sequence of tasks. It has to be minimized.

12. Average waiting time- is the average of time the tasks spend in the ready queue to be executed. It needs to be minimized.

13. Response time - the time passed since the release of the individual task until its execution is finished. It needs to be minimized.

14. Delay- the time between the maximum allowed response time of the task until its execution is finished. It has to be minimized.

3.2 Task Scheduling and Allocation in Distributed Systems

Various task assignment strategies have been developed over the years. They can be categorized as optimal and sub-optimal. The first method tends to reach the optimal solution, examples are graph theory, mathematical programming, and state-space search. On the other hand, sub-optimal methods try to achieve a good solution rather than a complete one. Sub-optimal methods, as well, can be split into approximate and heuristic solutions. Approximate methods is similar to the optimal methods in terms of the computational models used. On the other hand, heuristics tend to reach a good solution while utilizing different computational models.

Heuristics and meta-heuristics optimization methods are mostly applied in cloud computing based or fog computing based IoT [65], [66], [67], [68], [69], [70], [71], [72]. Other than IoT, heuristic optimization approaches are used in various fields for performing task scheduling such as: wireless sensor networks, cloud computing, fog computing, and embedded systems. It is a solving strategy that is problem independent, with fast convergence speed, and can search global solution. We tackle the most popular strategies:

3.2.1 Particle Swarm Optimization Based Task Scheduling

Particle swarm optimization (PSO) is a population-based search algorithm derived from the nature specifically the biological swarm behavior of populations. The algorithm preserves a set of possible solutions on the search path. It tends to shift from a group of points to another based on some rules during the one iteration. This shift is supposed to enhance the solution. It is easy to implement and

usually used to solve integer optimization problems. Some works applying PSO for task scheduling are [73], [74], [75], [76], [77], [78], and [79].

3.2.2 Genetic Algorithm Based Task Scheduling

Genetic algorithm (GA) is a population-based search algorithm as well, but derived from the reproduction behavior of populations. The algorithm tries to alter the candidate solutions for the purpose of reaching better ones. GA requires high computational costs and is more used for solving complex optimization problems [80], [81], [82], [83], [84].

3.2.3 Ant colony optimization Based Task Scheduling

Ant colony optimization (ACO) is a probability based technique inspired by the behavior of real ants which try to find the shortest path from a source of food to the ant nest with the aid of a chemical instance released on the moving path. Similarly, Ant colony method depends on recording the positions of previous solutions to be used later during upcoming iterations for reaching the final solution. ACO can adapt to changes despite that its time to converge is uncertain [85], [86], [87], [88], [89].

3.2.4 Simulated annealing Based Task Scheduling

Simulated annealing (SA) is an optimization algorithm for solving problems without constraints. It is derived from the physical process of heating to a high temperate and then decreasing the temperate gradually. The high temperate represents the initial solution. SA tends to generate a random points at each iteration based upon a probabilistic function. It can reach approximate global optimum, but is slow in reaching this solution [90], [91], [92], [93], [94].

3.2.5 Heterogeneous Earliest Finish Time

Heterogeneous Earliest Finish Time (HEFT) is a well-known heuristic based scheduling algorithm. It is intended to schedule a set of dependent tasks to a network of resources. HEFT base its solution based on computation and communication costs of the network. HEFT can achieve good solution

while being simple. At the same time, it is claimed to require additional load balancing mechanism [72], [95], [71], [96], [93].

3.3 Task Scheduling and Allocation in Cloud-Fog IoT Architectures

Promising trials are made to achieve dynamic allocation of tasks between the fog and cloud resources. By dynamic, we mean that the decision of where to execute the task is decided at runtime by management unit in the architecture based on the updated information about current tasks and available resources. This unit is sometimes called broker, control device, layer controller or even fog orchestrator. This problem is usually defined as task allocation/assignment or management of resource inside fog environment. The problem of task scheduling and allocation in the fog or the cloud is addressed multiple times outside the domain of the IoT such as [97], [98], [99], [100], [101], [102], and [103]. For example, the authors in [103] propose resource and task allocation methodologies for offloading smartphones applications supported by 5G network technology to the fog and the cloud. The purpose of their work is minimizing the computing and network energy. The type of tasks tend to be complex 5G stream applications. However, we focus on task scheduling and allocation when the fog and cloud computing are combined together in IoT architectures such that multiple heterogeneous sensors are utilized. These sensors generate independent small sized heterogeneous tasks with short delays.

In [104], the authors formulate the allocation problem as an Integrated linear programming (ILP) model for the objective of minimizing the delay in total when the resources are asked by the tasks. They point to the delay in their paper as the slot allocation time for the service on a fog layer. The model is presented in a combined Fog-Cloud (CFC) architecture where there are multiple fog layers between the things layer and the cloud layer. However, the authors calculate and represent the delay on a resource device by a time unit. They take no account of the diversity in tasks complexities nor the capabilities of the fog devices. In addition they only differentiate the fog devices in the first layer from the fog devices at the second layer by only adding a one more time unit delay for the devices in the second layer. They pay no attention to the differences in the communication cost based on the size of the data or the communication technology. Their work is extended in [105] to adapt multidimensional

knapsack modeling for the service allocation problem (MKP) while adding the energy consumption factor in the objective. Authors in [106] utilize the available resources whether storage or computation in fog environment and the cloud. They tend to enhance the performance of the system through the reduction of costs represented by the total delay and makespan for processing the tasks. Their objective in formulating and solving the problem is to maximize the possible task assignments to resources in a fog colony in a lower layer and avoid propagation of tasks to higher layer. From our point of view, their allocation methodology is missing the prioritization factor. Tasks generated later than their former miss the chance to be processed on the fog layer if their are no resources available as a result of their timing. In [107], the authors propose, through partitioning the architecture into three layers: Cloud, High Capacity Fog (HCF) and Local Fog (LF), two strategies for resource allocation. The authors solve Dynamic Service Execution (DSE) problem using First-fit and Random-fit algorithms. In first fit methodology, resources are allocated in the order LF, HCF, and cloud layer. In random fit, the layer where the service is allocated to one of its resources is chosen randomly. Indeed, despite that the authors consider first-fit as a prioritizing algorithm, we find it somehow a weak methodology. However, the authors take into consideration the assumption of the mobility of the computing resources in the fog. The authors evaluate their work through the measurement of the service/task response time, and the power consumed by the cloud devices only.

Trade-off approaches are also studied where multiple objectives are chosen for the allocation optimization problem. A trade-off approach is proposed in [108] in which the authors utilize Lyapunov optimization algorithm to build an optimization problem of unit slot. A monetary cost minimization objective as well as the minimization of applications response time in the architecture are their two objectives. The optimization problem is solved based on the assumption of zero cost in case that application is computed at fog layer, while utilizing cloud layer consumes communication and computation costs. Actually, we think that we cannot neglect the cost at the fog layer and it must be an important factor in the optimization problem. The authors in [72] propose Cost-Makespan aware Scheduling heuristic (CMAS) which is a strategy for task allocation on the fog devices as well as cloud devices for the purpose of reducing the latency and monetary cost of processing the required tasks. Their methodology is based upon the popular heterogeneous earliest finish time (HEFT) algorithm and

assumed the dependencies between tasks.

Both [108] and [72] share the same objective while having significant differences in the network and computation assumptions and proposed algorithms. For instance, as previously mentioned, [108] ignore the cost at the fog layer while [72] take the computation and communication at the fog devices into consideration. Furthermore, [108] assumes no interaction between the fog devices in addition to restricting the responsibility of the fog device in the execution of the tasks attached solely to IoT devices in its vicinity. This assumption is dissimilar to the one in [108] where any task can be executed on any computing device based on the optimization objective and constraints. This is the same assumption we are following in our work as well. In [109], the authors propose an estimation model of required computing resources to acquire the service(s). The authors considers other resources than the computations such as the storage and the bandwidth. Their estimation adapts the price cost of the resource and based on a probability function related to the customer (IoT device) requesting the resource. This probability function takes advantage of the history of the demands of the resources by this customer.

The previously mentioned approaches function with any IoT application, but an approach dedicated to healthcare, HealthEdge, is proposed in [110]. The authors build an optimization problem for managing the balance between task allocation on edge workstation or remote cloud datacenter. A heuristic solution is proposed for the objective of minimizing data traffic and completely utilizing the edge devices processing capabilities. The assignment of task in either edge queue or cloud queue is made by task priority determination and task latency estimation modules based on the emergence state. They lack reporting information about the communication protocols. A thorough search of the relevant literature yielded an article that take into consideration task criticality and its influence on the allocation methodology, which is [110]. Table 3.1 summarizes the main feature of aforementioned Cloud-Fog architectures.

Ref	Healthcare Application	Task	Allocation Scheduling	Technique	Objective
[14]	Vital sign monitoring	1) Vital sign measurement 2) Bayesian Belief Network Classification 3) ECG, EEG signal analysis 4) Alert generation - Temporal Health Index calculation	Static allocation, No scheduling	Fog role: tasks 1,2 Cloud role: tasks 3,4	Max. classification accuracy & Min. response time
[61]	Tele-health of speech motor disorder Tele-health of ECG monitoring	1) Dynamic time warping 2) Clinical speech processing chain 3) data compression 4) Data and bandwidth reduction	Static allocation, No scheduling	Fog role: tasks 1,2,3,4 Cloud role: storage & back-end analysis	Data reduction
[59]	Speech tele-treatment	1) Speech feature extraction	Static allocation, No scheduling	Fog role: task 1 Cloud role: long term analysis	Max. accuracy
[58]	Stroke mitigation	1) RSS detector 2) ADLs filtering 3) Data pre-processing 4) Non-linear analysis	Static allocation, No scheduling	Fog role: tasks 1,2 Cloud role: tasks 3,4	Min. response time, energy consumption & Max. accuracy
[57]	Fall detection	1) Fall detection algorithm 2) Threshold based gas detection	Static allocation, No scheduling	Fog role: tasks 1,2 Cloud role: long-term storage	Min. processing time
[60]	Assistance system to Chronic Obstructive Pulmonary Disease	1) Real-time monitoring of oxygen dose 2) Real-time estimation of effort 3) Tuning patient therapy to activity 4) Context information collection and processing	Static allocation, No scheduling	Fog role: tasks 1,2 Cloud role: not declared	Min. incidence readmission rate
[110]	Generic, Use-cases: monitoring system for persons with dementia	1) QRS wave detection 2) RR variability extraction 3) De-noising 4) Anomaly detection 5) Other medical analysis	Dynamic	Heuristic	Minimize network traffic load & total processing time

Chapter 4

Security and Privacy for IoT Healthcare

In Internet of things (IoT), where heterogeneous devices and network technologies are embedded and interconnected together to the Internet, security and privacy are important demands. It is required to guarantee the protection of the IoT devices having Internet accessibility, in addition to protecting the connection between the IoT devices themselves. This in turn leads to the protection of the data transferred which contains vital medical data in healthcare applications.

This chapter surveys security and privacy issues in Internet of things (IoT). We discuss security and privacy requirements, in addition to presenting existing protocols and mechanisms employed in IoT.

4.1 Security Requirements

Multiple security and privacy requirements must be taken into consideration when considering an IoT environment

- Data confidentiality: Intends to protect the data against unauthorized access or revelation whether planned or accidental. Confidentiality is considered to protect the secrecy of the data such that it is disclosed only to the intended recipients, and according to the rules and regulations.
- Integrity: Is to protect data from being altered or modified improperly by a third party. Furthermore, it ensures and maintains the accuracy of transmitted data.

- Availability: Means the ability to access the data at anytime such that critical data is continuously available when needed. Furthermore, it is required to prevent attacks that disrupt the availability of IoT devices and hence, interrupt the services provided by these devices.

- Privacy: Is a requirement that controls how the personal information is collected and used. It constitutes and important demand in healthcare applications as medical sensors collect and transmit patient's medical data .

- Authentication: Data authenticity is a valuable requirement in security and privacy. It is the method to validate the communicating parties before and during the exchange of information. Usually, mutual authentication is required in a communication channel which means that the sender has to prove itself to the receiver and vice versa. Authentication

- Authorization: Defines the access privileges that given to the users/devices to obtain or alter the data.

- Forward secrecy: Is the requirement the forbids a device that leaves the network from obtaining the future key or data exchanged.

- Backward secrecy: Is the process that forbids a new device joining the network from obtaining past key or data previously exchanged.

  To summarize, authentication ensures that the communication is performed only between the authorized parties, whereas confidentially and integrity ensure the protection of the data itself in such a way that they all achieve privacy of patients and the security of information respectively.

4.2 Attacks

Internet of things (IoT) is a collection of smart devices connected to the Internet, in addition to communication together through one or more of wireless sensor networks (WSN), wireless local area network (WLAN), wireless personal area network (WPAN), wide area network (WAN), or cellular networks. Hence, it is prone to the most popular network attacks. Adding to that the constrained nature of some

of IoT devices, where they suffer computation and storage limitations. This in turn poses additional vulnerabilities due to their resource limitation that forbids them from implementing powerful security mechanisms.

4.2.1 Common IoT attacks

In this section, we list the most popular attacks in IoT

1. Collusion attack: is the process of combining numerous copies of data to generate new one. This can be achieved through applying replacement, averaging, or even linear combination methods.

2. Insider attack: is an attack where an authorized person maliciously attacks a device or a network on the system.

3. Eavesdropping: in this attack, an attacker listens to the network continuously for the purpose of stealing the sensitive information during its transmission over the unsecured network. Eavesdropping is also called sniffing or spoofing attack. This attack leads to leakage of data which threatens the application.

4. Impersonation: it is when an adversary pretends to be an authorized device to obtain illegitimate access to critical data. An adversary in this way is considered a rogue IoT node. An example is when a malicious access point assembles the produced data from connected IoT devices in order to misuse it. In other cases, this rogue node provide neighboring nodes with malicious data.

5. Denial Of Service (DOS) attack: is commonly a network or device attack where perpetrator floods the network or device respectively with traffic to disrupt its service or functionality.

6. Sinkhole attack: is an attack where an adversary node attracts the traffic through forge routing. Next, this attack can be combined with other attacks such as selective forwarding or eavesdropping to modify or damage the data.

7. Jamming attack: is a sensing layer attack similar to denial of service (DOS) attack where an attacker jams the radio signals of the device with flooding signals such that the device cannot

communicate. Jamming attack works with different radio networks whether cellular, WiFi, or other.

8. Selective forwarding attack: is a sensing layer attack as well. In this attack, an adversrary node refuses to forward selected packets. In this way, the rate of packet loss is increased and hence, the operation of the network is corrupted affecting the quality of service.

4.2.2 Cloud-Fog threats

In Cloud-Fog environments, the fog nodes and the cloud nodes are the devices applicable for processing and storage on behalf of the resource constrained IoT devices and sensors. However, fog computing and cloud computing environments are prone to various attack. Here, we list some of them:

1. Compromise of poorly scattered fog node: For instance, an adversary can compromise gateway that serves as a fog node. As the fog nodes process or forwards the data from multiple IoT resource constrained devices and sensors, compromising a fog node can reveal sensitive or private information such as identity, location, or even usage or mobility pattern.

2. Cloud node malicious insider: Cloud nodes are where the long-term processing is performed, and permanent data is stored. Cloud nodes are remotely provided and accessed which makes the process of securing these data a challenging task. The threat to the data at the cloud can be an intentional by a malicious insider, or even accidental such as loss or improper altering or delete without recovery plan.

   In addition to the above threats, cloud and fog raises virtualization challenges. The processing performed at the fog node or cloud node is usually executed through virtualization. During virtualization, fog or cloud nodes host virtual machines (VMs) while sharing resources such as processor/cores, memory, cache, network interface card. Hence, an attach can be issued between VMs on the same device or different devices, but sharing resources. Next, we present the most common threats in virtualized environments such as the cloud or fog.

3. Side Channel Attack (SCA): A side channel is a channel that exists in the hardware, but is hidden. Examples on SCA are the cache channel between two processors or two virtual machines. Side channel attack exploits the cache shared between virtual machines (cross-VMs) to gain unauthorized access to the information. Also, side channel can also be an implementation to improve the utilization of the cache resource such as hyper threading.

4. VM migration attack: virtual machine migration refers to the process of moving a VM physically from a host to another due to upgrade or maintenance reasons.

    - Migration flooding: In this attack, the adversary can gain unauthorized access to the module controlling the migration process. In this way, it can move the virtual machine to resource limited device where the processing and storage requests are flooded without response.
    - False resource advertising: In this attack, the adversary can compromise a victim fog or cloud server and ask for offloading the virtual machines to it as having enough resources.
    - Spoofing attack: This attack is the previously described network attack, but the attacker listens to the network lines where the virtual machine is transmitted to gain illegitimate access to information.

4.3 Existing protocols and solutions

This section discusses the different security protocols applied in IoT. Taking into consideration the resource constrained environment and the architecture of IoT healthcare systems, we propose the common protocols applied in IoT as well as lightweight cryptography which is specifically designed to constrained IoT devices.

4.3.1 Low-Level Solutions

This type is considered with solution for attacks happening at the physical and data link layer such as jamming, denial of service (DOS), and eavesdropping attacks. They are mainly concerned with

the detection of the attack and using a cryptographic algorithms to defend these attacks. Multiple solutions are proposed in this area such as [111], [112], [113], [114], [115], [116],[117], [118] and [119]. An example in [115] when the authors try to locate an adversary that is located near the device attacked. They apply enhanced secrecy utilizing synthetic noise. Similar idea is utilized in [116] where the idea of artificial noise is adapted to detect the adversary. In [117], the authors propose a framework that performs a configuration between the communicating devices utilizing data rate in its minimum value to protect against attackers. The authors in [118] discuss the interference in networks and models of attack at low-level, in addition to proposed solutions to this level such as collision detection, cryptographic authentication.

Despite that security solutions at a low-level target to detect and prevent the collection, manipulation of data, these methods could fail to function as required. In this case, strong cryptography implemented using data encryption, efficient key management is a must in IoT security. In this way, it is hard for the adversary to understand the acquired data and hence, to forge or alter it. The next two subsections present data encryption methods and key management methods in IoT networks.

4.3.2 Data Encryption

Data encryption algorithms can be categorized into

- Hop by Hop Data Encryption: The idea behind this type is that each node and its neighbors share different keys than the other devices in the network. In [120], the authors propose SecureDAV, an elliptic curve based crtyptographic mechanism for resource constrained networks. Their idea is based on developing a secret key for a cluster which is shared between the resource constrained devices in the cluster. Finally, a base station performs the authentication on behalf of the resource constrained devices using the assembled key. The authors in [121] present a forward authentication mechanism that is used to aggregate the data in a secure way. The mechanism performs aggregation operations such as calculating the median, minimum, maximum, counting, averaging in a secure way. The authors in [121] propose a protocol that protects the aggregated data from a malicious node attack that tries to corrupt the data. This is done through sharing a secret key between the aggregating device and a home server where the data is finally sent. The

aggregator sends a message authentication with the data to the home server through the sensor node. Due to their limited capabilities, the sensor devices doesn't store key, but instead they message authentication code (MAC) of the communication nodes only. This method is prone to vulnerability risk at each node when the data is decrypted.

- End to End Data Encryption: In this mechanism, the data is not decrypted between the communicating devices on the path from the source to destination. Despite that it forbids the data from being revealed at intermediate nodes, this method is easy to attack if the key is compromised at any node. As an example, in [122], the authors focus on providing security during the process of data aggregation. They utilize homomorphic encryption which is a methodology to process the encrypted data without decryption it. Hence, in [122], the sensing devices can assemble the data aggregated by themselves in addition to the cipher received from their neighbors by performing arithmetic and statistical computations in their encrypted format. The sensing devices then forward the processed data to their next hop until reaching the end use. A similar approach is proposed in [123] where the authors apply PH which stand for privacy homomorphism to perform two operations: averaging and movement detection. Another works are presented in [124], [123], [125], and [126].

4.3.3 Key Management

In order to encrypt the data being shared between the communicating parties, there must be an efficient key management protocols to establish the keys used in the encryption. Key establishment protocols can be categorized into pairwise key establishments and group-wise keys. Pairwise key establishment protocols are then classified into symmetric key and asymmetric key.

- Symmetric key establishment protocol: in this category, the communicating devices share a pre-shared key. Symmetric key protocols are energy preserving methods due to less computations and causes less overhead. However, it is non scalable when the number of devices increases. Examples on symmetric key pre-distribution algorithms are [127], [128], [129], [130], [131].

- Asymmetric key establishment protocol: this protocol is a real-time method that allow the com-

municating devices to exchange a pair of keys to authenticate. Asymmetric protocols are more secure and scalable, usually applying elliptic curve cryptography (ECC) or RSA. They can be extended to more complex where certificates are exchanges beside the keys [132], [133], [134], [135], [136].

- Group-wise key establishment: In this category, a single key is shared between multiple communicating devices (a group). Multiple protocols implement this method following different methodologies such as [137], [138], [139], [140], [141]. In [137], a pre-shared key based group key establishment is proposed. The key is shared between a group of communicating devices in the same group, not only between a pair of devices. In [138], the protocol formulates a tree graph describing the relation between a number of keys and a number of devices in the network. This protocol is appropriate for small scaled networks where the tree can be simply constructed and implemented. A method in [139] is proposed where the keys are not pre-established, but instead they are created at run time by the users or devices. An approach where the group keys are changing during the life time of the network is proposed in [140] called Veltri's. In Veltri's, a differnt key is generated for each time slot and shared between the members of the group. The authors in [141] propose DBGK protocol which share a similar idea with [140] whereas an entity called AKMS (area key management server) is responsible for creating and publishing the keys based on the entry/exit of devices to/from the group.

4.3.4 Lightweight Cryptography

As IoT usually comprises IoT devices with limited resources, a research direction that targets the development of lightweight cryptography methods. Many cryptographic algorithms are recently developed to be less in computation complexity and short in footprint to adapt with the constrained IoT devices. These algorithms are called lightweight cryptography. lightweight cryptography also aims at providing solutions that preserve energy efficiency, and are capable of presenting high level security. Next, we present some of these security algorithms:

- Advanced Encryption Standard (AES)-128: AES-128 is a lightweight version of the known Advanced Encryption Standard (AES) block cipher protocol where the key size is 128 bit. Hence,

it is adequate to constrained IoT devices due to small storage required in addition to less complexity regarding the operations performed on 128 bit instead of 256 bit standard AES. Despite that, it is a strong security protocol [142].

- Data Encryption Standard Light(DESL): DES-56 is a lightweight version of the known Data Encryption Standard (DES) block cipher protocol. It is less secure than AES, but can be fit with very constrained devices and hence, allows for providing the minimum protection for devices in IoT [143].

- Grain 128-a: Grain is an example on lightweight stream cipher security protocol for IoT. The key size is 128 bits which provides a strong while being simple and fast due to its type as a cipher stream protocol.

- Internet Key Exchange v2 (IKEv2): IKEv2 is a lightweight key management protocol for resource constrained IoT devices. It is a lightweight version of IKE used by IPSec protocol implemented at the link layer of the IP protocol stack [144].

- Diet Host Identity Protocol (Diet HIP): Diet HIP is a lightweight version of HIP implemented at the link layer as well. It is based on the host identification feature [145], [146], [147].

- Extensible Authentication Protocol (EAP): EAP is a lightweight authentication protocol implemented at the data link layer. It suits IoT in the way that proposes low overhead while being flexible. EAP allows for re transmission and hence, is able to detect errors and deliver data to its destination [148].

- Datagram Transport Layer Security (DTLS): DTLS is a lightweight version of the Transport Layer Security (TLS) protocol to protect data privacy at the transport layer. DTLS uses UDP to adapt with the constrained capabilities of IoT devices while TLS uses TCP [149].

Chapter 5

Proposed Cloud-Fog based Architecture for IoT Healthcare

In this chapter, we propose a Cloud-Fog based comprehensive architecture that can embrace multiple healthcare scenarios and able to adapt dynamically with the context and status of the patients. We tend to tackle the non-generic implementation in existing Cloud-Fog architectures in healthcare. To clarify more, existing researches design and implement fixed infrastructures that are specific to certain medical cases e.g. heartbeat monitoring which receives specific vital signs, etc. Hence, supporting different medical cases or even updating the current medical case require restructuring the whole architecture. We hereafter present a healthcare architecture that is Cloud-Fog based and generic.

The architecture proposed is generic in the way that it can handle multiple therapeutic condition e.g. different healthcare medical cases (applications) and different number of patients. In addition to the generic implementation that supports diverse medical cases, our proposed architecture allows the dynamic distribution of healthcare tasks between the fog computing as well as cloud computing through an implementation of a scheduling and allocation approach to balance the distribution of tasks processing among the cloud devices and the fog devices. Different virtual run-time environments defining the software of each task are employed on the computational devices in the form of SaaS (software as a service). In our contribution, we utilize specific features for the tasks in the healthcare application. This chapter is composed of three sections: section 5.1 presents the proposed architecture, section 5.2 describes the system model including features of the tasks, and proposed scheduling solution. Finally, section 5.3 evaluates the proposed approach.

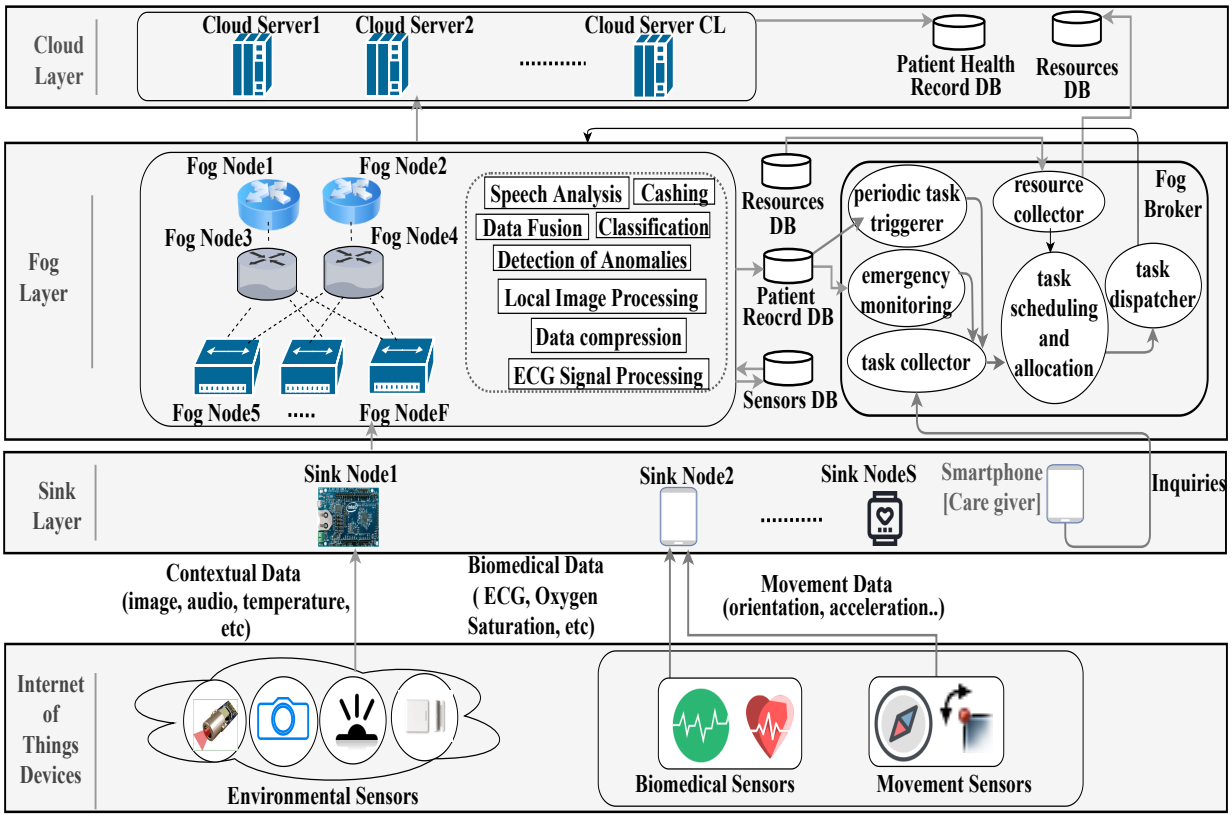


Figure 5.1: Proposed IoT architecture for healthcare.

5.1 Proposed Architecture

Our architecture assumes an environmental context of a hospital or a retirement home which encompasses a user context of multiple mobile patients who are able to perform diverse set of activities. Our architecture is not restricted to specific medical cases as presented in the previous section. It can support patients with different diseases for example patients with Parkinson disease who need speech analysis process in addition to patients with heart problems who need ECG signal processing, etc. An extension of the scripts describing tasks to be processed is added in task triggering modules as will be described next. Furthermore, based on the algorithms implemented, our architecture enables dynamic distribution of health tasks between the fog and the cloud devices in real time in contrast to static off-line approaches in the literature. The proposed architecture is shown in Fig. 5.1 and consists of

four layers as follows:

1) **Things layer (IoT devices)**

The things layer contains all sources of data needed for the healthcare system. We describe two sets of sensors in this healthcare architecture; set A and set B. Set A contains the environmental sensors connected through wireless sensor networks (WSNs). These sensors are static such as Infrared (IR), cameras, light and temperature. Set B involves wearable sensors that generate two types of sensing data: activity monitoring and biomedical (BIO). The activity sensors are physical body sensors that are used to infer the motion such as accelerometer and gyroscope. Examples of BIO sensors are the heart beat, ECG, and glucose sensors.

2) **Sink layer**

The sink layer contains the data aggregating devices such as the micro-controllers, smart watches, and mobile devices. These devices are responsible for: *1) Data acquisition;* the sink device acquires and collects the raw data generated from the data sources in the things layer, *2) Data forwarding;* the sink device forwards the collected data to the fog layer for further processing, *3) Inquiries forwarding;* the sink device forwards the inquiry requests to the fog layer. Inquiries are on-demand requests issued by the end users of the system including the care givers and the patients themselves to obtain health information. The inquiries are issued using mobile devices connected to the fog layer - mainly the fog broker. Examples include when a patient wants to know his updated pills timetable assuming the timetable is updated periodically according to the patient's health status during the last 24 hours. Another example is when a doctor wants to get data log on the last 100 heartbeat readings of a patient, *4) Limited processing;* the sink device helps in the processing of some tasks according to their processing capacity and if this will lead to the global objective of the allocation. The sink devices are connected to the fog layer where the data is received and stored in the sensors database.

3) **Fog layer**

The fog devices are devices with high computation, communication, and storage capabilities, and

are capable of performing diverse processing operations. Each fog device contains its internal database used for the computation and storage of its allocated tasks. Beside the internal databases, three global fog databases are maintained in this layer: *1) Resources database* where the processing capacity, utilization, and storage of each processing device are reported, *2) Sensor database* where all the sensing data reported from the things layer is stored and updated, and *3) Patients record database* for storing the results of the analysis operations. The fog devices are responsible for performing the computations of the following four sets of tasks: *data analysis, critical analysis, critical control* and *context management* [150]. Critical health analysis and critical control tasks have relatively higher priorities. A fog broker is responsible for querying information from the *Resources database* to manage the allocation of described tasks on those resources. An extension can be made to include more than one fog broker device if needed and if there are cases that allow for that.

Fog broker It includes the following modules: *a) Periodic Task Trigger.* This module is responsible for releasing periodic tasks. We assume a set of periodic health tasks with different frequencies and deadlines which are varied based on the current configurations and requirements of the environment (number of medical cases, current status of the patients health, etc). *b) Emergency Monitoring.* This module continuously monitors the patients' status by accessing the *patient record database*. Through the continuous checking of the results of the analysis operations performed, this module can detect the emergency cases. Next, proper emergency actions are handled whether notifying the care givers or triggering critical control tasks which are passed to the task collector module to be performed. *c) Task Collector.* This module collects all the tasks needed to be processed from the periodic task trigger and emergency monitoring modules, and inquiries from the patients and the caregivers. It establishes a set of structures containing information and features of the tasks. *d) Resource Collector.* This is the module that receives information about the available resources on the fog layer. It maintains the utilization of the fog devices as well as the updated capacities of these devices. *f) Task Scheduling and allocation.* This module is responsible for the formulation and setup of the task allocation process for the fog and cloud devices. This module mainly implements a software routine for an optimization technique to achieve an optimal solution for minimizing the latency of performing the health tasks. Heuristic approaches can reach approximate solutions in a faster way than optimization techniques that search for

optimal solutions [151][152]. The output of this module is a scheduled mapping between the tasks to be performed and their allocated fog or cloud devices. *e) Task Dispatcher.* This module is responsible for contacting the fog and cloud devices, assigning them their allocated tasks.

4) **Cloud layer**

The cloud layer contains resource devices on the cloud for performing part of the tasks that are not performed in the fog layer; usually time unconstrained tasks. This layer contains a permanent database for keeping the patient's health analysis results to be used by the care givers.

5.2 Task Scheduling and Allocation

This section describes an approach for task scheduling and allocation that is implemented in the fog broker. The approach achieves a balanced and effective task scheduling and allocation to healthcare tasks. Our objective is to minimize the latency of healthcare tasks and ensure that critical tasks meet their deadlines.

5.2.1 System Model

The input to the task scheduler and allocation module includes a set of independent input tasks ($\mathcal{U}$) and a set of fog devices ($\mathcal{V}$), which are represented as a bipartite graph, and a set of cloud devices ($\mathcal{W}$), as shown in Fig. 5.2. The bipartite graph links, through an edge, each task represented as a vertex in the set $\mathcal{U}$ with one or more fog devices represented by a vertex in the set $\mathcal{V}$, where $\mathcal{U}$ and $\mathcal{V}$ are two disjoint and independent sets. The existence of the edge means that the data related and needed by that task to be executed is resided in the internal database of that fog device. So, an edge describes the locality of the data in a fog device. This locality comes from the layered architecture where the sensors are connected to their upper layer sink and gateway devices. Hence, the data of these sensors resides initially on that sink and gateway devices before they are transmitted to the global database in the fog layer or in the cloud layer. Each task is defined by a set of six features:

1. $TaskID$ which is a number that uniquely identifies the task.

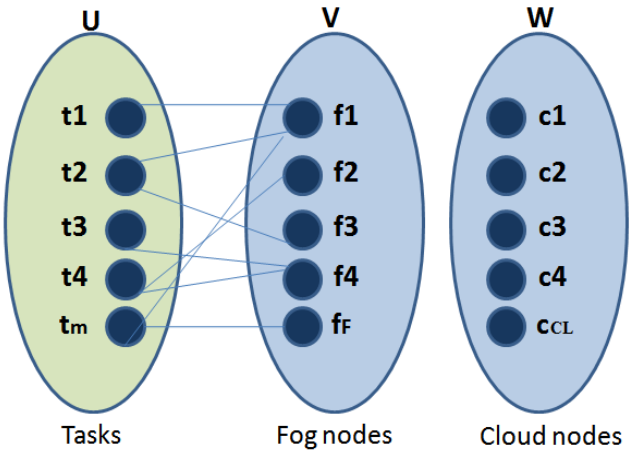


Figure 5.2: Input to the task scheduler and allocation module.

2. $Class\ of\ the\ task(Emergency\ Level)$ the class is a number that classifies the task type into four classes of three categories according to their priorities. These classes are uniquely defining the emergency level of the medical measurement among the heterogeneous measurement and different medical cases.

3. $Size\ of\ the\ measurement\ data$ that the task will work on and which depends on the segment collected from the corresponding sensor(s).

4. $Maximum\,Response Time$ while in real-time applications there is a demand for quick processing of the tasks, in healthcare scenarios, each task has a diverse maximum time that the system has to execute the task within.

5. $Task\ complexity$ complexity of the task is usually defined in the number of cycles it takes the processor to execute the task. We can refer to the complexity of the task as the number of instructions to be processed for this task which is defined in terms of MI.

6. $Set\ of\ Sensors$ related to this collected task.

The objective of the proposed approach is to choose the best fog or cloud device to allocate a specific task to, and to schedule all the tasks allocated so that latency can be minimized. The proposed approach is divided into two phases. The first phase is responsible of prioritizing each task according to

its emergency level and its required maximum response time. The methodology used for this phase relies on applying weighted sum method (WSM) decision making algorithm. WSM is a robust multi-criteria algorithm well-known in the decision theory. We choose this method because of its flexibility that enables us to variate the weights assigned to each feature of the task according to the situation [153]. The second phase proposes and implements a task scheduling and allocation algorithm, Modified BAR, based on the BAlance-Reduced (BAR) algorithm [101]. Table 5.1 enumerates a group of notations that are utilized throughout chapters 5 and 6.

5.2.2 Ranking

The WSM algorithm is implemented to evaluate and rank the tasks to be executed in terms of task classification and maximum response time according to Eq. 5.1:

$$T_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij} \qquad for\ i = 1, 2, 3, ..., m \tag{5.1}$$

$T_{t_i}^{WSM-score}$ refers to the total importance of task t_i. The higher $t_i^{WSM-score}$, the earlier the task t_i to be scheduled. w_j is the relative weight of importance of the criterion c_j, a_{ij} is the performance value of task T_i when it is evaluated in terms of criterion c_j and m is the total number of tasks. We set $n = 2$ to exploit two significant weighting criteria: $Class_{t_i}$ and inverse of $MaxR_{t_i}$. We choose four classes to categorize the tasks. Section 5.3.1 clarifies more details regarding classes of the tasks. Maximum response time is represented in order of milliseconds, seconds or minutes. The importance of the task is inversely proportional with the value of the maximum response. We choose $w = 0.5$ because we assume that the prominence of the task is equally shared between the two criteria. The ranked list is scanned and the tasks are scheduled and allocated according to the second phase.

5.2.3 Modified BAlance-Reduced (MBAR) algorithm

MBAR is a heuristic algorithm which means it is fast and feasible; thus, it can provide a quick scheduling solution for our problem. The main idea behind MBAR is its data placement dependability such that the algorithm firstly tries to place the tasks on the processing devices that contain their data streams as a preliminary phase. The motivation behind this step is to reduce the transmission costs spent to

Table 5.1: Notations

Notation	Description
α	The interval between consecutive executions of allocation algorithm
β	Allocation outcome
$\mathcal{U}$	A set describes input tasks independently related, of length m
t_i	Task i $\in \mathcal{U}$
$Class_{t_i}$	Classification of task t_i
$MaxR_{t_i}$	Maximum allowed response time to execute task t_i
Θ_{t_i}	Complexity of task t_i
D_{t_i}	Volume of collected data required to process task t_i
$\mathcal{V}$	A set describes fog devices
$\mathcal{W}$	A set describes cloud devices
P	A processing device (cloud or fog)
N	The overall cloud and fog processing devices number
L_P^{init}	Pre-allocation initialized workload of P
L_P^{fin}	Post-allocation final workload of P
$L_P(t_i)$	Load of processing task t_i on computation device P
$e(t_i, P)$	Execution time of task t_i on P
$t(t_i, P)$	Time needed to transmit data required by task t_i to processing device P
C_P	Computational capacity of P
BW_P	The bandwidth of link between P and $sensors\ DB$

transmit the data to remote processing device. The subsequent steps of the algorithm tend to best utilize the remote processing devices (whether in the fog layer or the cloud layer in our architecture) to offload some of the tasks basically allocated in the first step. The main objective of the proposed algorithm is to minimize the total length of the scheduler generated.

To solve the aforementioned problem model, two steps are performed: Balanced step and Reduce step. These steps helps us to reach the allocation decision which means whether the processing of the tasks is performed on a cloud or fog device. Algorithm 1 presents the pseudocode of the proposed scheduling and allocation approach MBAR.

1) In **Balanced step**, we allocate a task t_i on a processing device considering task data locality or in other words, where the collected data resides. We select a processing device P which has the minimum processing load to allocate a task t_i according to Eq. 5.2.

$$\beta(t_i) = \min_{P \in \mathcal{V}} L_P \tag{5.2}$$

The load represents the finish time after the execution of that task and can be calculated following Eq. 5.3:

$$L_P = L_P^{init} + L_P(t_i) \tag{5.3}$$

where L_P^{init} is the initial workload of a processing device P. This value is zero at the starting of the configuration of the environment and updated continuously, and $L_P(t_i)$ is the total time to process task t_i on that processing device and calculated according to Eq. 5.4.

$$L_P(t_i) = e(t_i, P) + t(t_i, P) \tag{5.4}$$

$e(t_i, P)$ is the execution time of task t_i on processing device P and $t(t_i, P)$ is the transmission time for transferring the data needed by task t_i to the processing device P. In balanced step, $L_P(t_i)$ is calculated taking into account the execution time only. Hence$t(t_{i,P}) = 0$. The execution time $e(t_i, f)$ of processing a task t_i on processing device P is calculated according to Eq. 5.5

$$e(t_i, P) = \frac{\Theta_{t_i}}{C_P} \tag{5.5}$$

The complexity of task t_i is obtained from the task collector module and is represented in MI (millions of instructions). The processing capacity of the computing device P is obtained from the resource collector module and is represented in MIPS (millions of instructions per second).

2) The **Reduce step** starts directly after the balanced step and iteratively reallocate some of the tasks

on remote processing device to enhance the schedule length. Remote processing device means a cloud device or a fog device that doesn't store the task data locally. Firstly, we select processing device that holds the maximum workload using Eq. 5.6.

$$L_{Pmax}^{fin} = \max_{P \in \mathcal{V}} L_P^{fin} \tag{5.6}$$

Secondly, we choose a task to reallocate it. We implement three different strategies to choose the task to be reallocated;

A. The first strategy selects a random task, whereas the second and third strategies are based on the nature of the application (Random).

B. The second strategy selects the task with the highest class (Classification).

C. The third strategy chooses the task with the highest maximum response time (MaxResponse).

Next, we calculate the expected makespan after the reallocation step

$$Makespan_{expected} = L_{P_{max}}^{fin} - L_{P_{max}}(t_{reallocate}) \tag{5.7}$$

so that task $t_{reallocate}$ can be totally allocated to another remote processing device. Following, an iteration over all the other remote processing devices is started to find P that can achieve expected makespan. Eq. 5.4 is used again to calculate $L_P(t_i)$ where $e(t_i, P)$ is calculated according to Eq. 5.5, and transmission time $t(t_i, P)$ is calculated following:

$$t(t_i, P) = D_{t_i}/BW_P \tag{5.8}$$

where D_{t_i} is the size of the measurement data obtained from the task collector, and BW_P is the bandwidth of the link connecting the sensor database in the fog layer with the processing device P.

At this step, MBAR reaches one of three cases: an update in the allocation β when the expected makespan is achieved. Otherwise, if the expected makespan is missed, but new makespan is reached which is better than the old makespan, an update in β is performed as well. or the termination of the reallocation step due to the old allocation being the best. For the reallocation, new iterations are started

to locate P that fulfills Eq. 5.2 such that $e(t_i, P)$, $t(t_i, P)$ are calculated according to Eqs. 5.5 and 5.8 respectively [101], [154]. The makespan of the scheduler is calculated as

$$Makespan_\beta = \max_{P \in \mathcal{V} \cup \mathcal{W}} L_P^{fin} \tag{5.9}$$

where L_P^{fin} is

$$L_P^{fin} = L_P^{init} + \sum_{t_i : \beta(t_i) = P} L_P(t_i) \tag{5.10}$$

Hence, the final workload of the device P is the initial workload of a computing device, which describe the readiness of the device to process new task, and is accumulated according to the number and complexities of tasks allocated to it. Note that the actual workload of the processing device is the processing time after allocating a new task. The final workload of the processing device is the end processing time after all tasks are allocated.

5.3 Performance Evaluation

This section discusses the experimental setup and evaluation metrics. The configuration details of the fog devices and cloud devices are shown in tables 6.4 and 6.3 respectively. The simulations are carried out using iFogSim [155] which is a simulator for modeling customized fog computing environments with large number of fog devices and IoT devices such as sensors, actuators, routers, etc. In the simulation environment, we set two sink devices connected to each fog gateway. All the simulation results are averaged over 10 runs.

5.3.1 Healthcare Tasks

In our simulations we used healthcare tasks information collected and inferred from the reviewed articles proposing IoT healthcare applications [150]. The tasks are categorized into five classes as assembled by the authors: *context management, critical control, critical analysis, data analysis* and *data collection*. *Data collection* is performed at the sink layer in our proposed architecture. We order the classes as shown in table 5.4. Configuration details of the tasks features are shown in table 5.5.

Algorithm 1 Pseudocode for MBAR

Define $\mathcal{U}$ as a set that describes input tasks
Define $\mathcal{V}$ as a set that describes fog devices
Define $\mathcal{W}$ as a set that describes cloud devices
Define $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ as bipartite graph of sets $\mathcal{U}, \mathcal{V}$
Define (t_i, d_i) as $(task, IoTdevice)$ pair

Phase 1 – Local allocation

procedure BALANCE
Input: Bipartite graph $\mathcal{G}$
Output: Allocation β
  for each task t_i (represented by vertex $v1 \in \mathcal{U}$) **do**
    for each processing device P (represented by vertex $v2 \in \mathcal{V}$ such that $\exists\, e(v1, v2) \in \mathcal{E}$) **do**
      $L_P(t_i) \leftarrow CalcWorkload(\theta_{ti})$
    end for
    $\beta(t_i) \leftarrow MinWorkload\{L_P(t_i)\}$
    $L_P^{fin} \leftarrow L_P^{fin} + L_P(t_i)$
  end for
end procedure

Phase 2 – Remote allocation

procedure REDUCE
Input: Bipartite graph $\mathcal{G}, \mathcal{W}, \beta$
Output: Allocation β, Makespan
  $P_{max} \leftarrow MaxWorkload\{L_P^{fin}\}$
  $Makespan_{old} \leftarrow L_{Pmax}^{fin}$
  $\beta_{old} \leftarrow \beta$
  do
    $t_{reallocate} \leftarrow Choose\ according\ to\ strategy$
    $Makespan_{expected} \leftarrow L_{Pmax}^{fin} - L_{Pmax}(t_{reallocate})$
    $L_{Pmax}^{fin} \leftarrow L_{Pmax}^{fin} - L_{Pmax}(t_{reallocate})$
    $\beta \leftarrow \beta - \beta(P_{max})$
    for each $P \in \mathcal{V}, \mathcal{W}$ such that $P \neq P_{max}$ **do**
      $L_P(t_{reallocate}) \leftarrow CalcWorkload(D_{t\_reallocate}, \theta_{t\_reallocate})$
    end for
    $\beta(t_{reallocate}) \leftarrow MinWorkload\{L_P(t_{reallocate})\}$
    $L_P^{fin} \leftarrow L_P^{initial} + L_P(t_{reallocate})$

Algorithm 1 Pseudocode for MBAR (continue)

$P_{max} \leftarrow MaxWorkload\{L_P^{fin}\}$
$Makespan \leftarrow L_{Pmax}^{fin}$
if $Makespan \leq Makespan_{expected}$ **then**
$\beta_{old} \leftarrow \beta$
$Break$
else if $Makespan_{expected} < Makespan$ && $Makespan < Makespan_{old}$ **then**
$Return$
else
$\beta \leftarrow \beta_{old}$
$Makespan \leftarrow Makespan_{old}$
$Return$
end if
while
end procedure

Table 5.2: Configurations details for fog devices

Parameter	**Value**
Number of fog devices	22
Processing capacity	[10, 100] MIPS for sink devices [100, 500] MIPS for others
RAM	[512, 1] GB
Storage	4 GB
Bandwidth	1024 Mbps

5.3.2 Evaluation Metrics

We evaluate the proposed architecture and algorithms according to four metrics: makespan, miss ratio, average delay, and cost per execution.

- Makespan: which defines the total length of the schedule and it is calculated using Eq.5.9. where $L_{fin}(P)$ is the final processing load of processing device P.

Table 5.3: Configurations details for cloud devices

Parameter	**Value**
Number of cloud devices	25
Processing capacity	[250, 1500] MIPS
RAM	40 GB
Storage	11 TB
Bandwidth	10 ,100, 512 Mbps
Cost per time unit	3 $

Table 5.4: Healthcare task classes

Class	**Description**	**Emergency Level**
Critical analysis	Data analysis of vital sign conditions	High-Level
Critical control	Processes for controlling actuators	High-Level
Context management	Context monitoring for algorithm such as location	Meduim-Level
Data analysis	Long-term deep medical analysis	Low-Level

Table 5.5: Configurations details for tasks

Parameter	**Value**
Number of tasks	300
Complexity	[1, 3000] MI
Data size	[100, 500] MB
Max response time	[15 sec, 30 min]
Sampling rate	[20, 5000] Hz

- Miss ratio: is the percentage of tasks that misses the maximum response time and it is calculated

as in Eq. 5.11.

$$MissRatio = \frac{No.\ of\ tasks\ missing\ MaxResponse(n)}{Total\ number\ of\ tasks} \tag{5.11}$$

A task t misses max response time if the condition in Eq. 5.12 is true.

$$Trig(t) + MaxResponse(t) < CompletionTime(t) \tag{5.12}$$

where $Trig(t)$ is the time when the task is triggered by periodic triggerer, emergency monitoring modules, or inquired by the users.

- Average delay: is the average of the times that exceeded the max response time for all the tasks and it is calculated as in Eq. 5.13.

$$AverageDelay = \frac{\sum_{j=1}^{n} Delay(t_j)}{No.\ of\ tasks\ missing\ MaxResponse(n)} \tag{5.13}$$

where t_j is a task that misses its maximum response time.

- Cost per execution which is the total execution cost for using cloud resources and it is calculated using Eq. 5.14.

$$CostPerExecution = cPt \times \sum_{\forall C \in CL} L_f(C) \tag{5.14}$$

where cPt is the processing cost per time unit, $L_f(C)$ is the final processing load of a processing cloud device C, and CL is the total number of cloud devices.

We investigate the performance of the aforementioned parameters with varying the number of tasks and the number of cloud devices with respect to three different task selection strategies implemented.

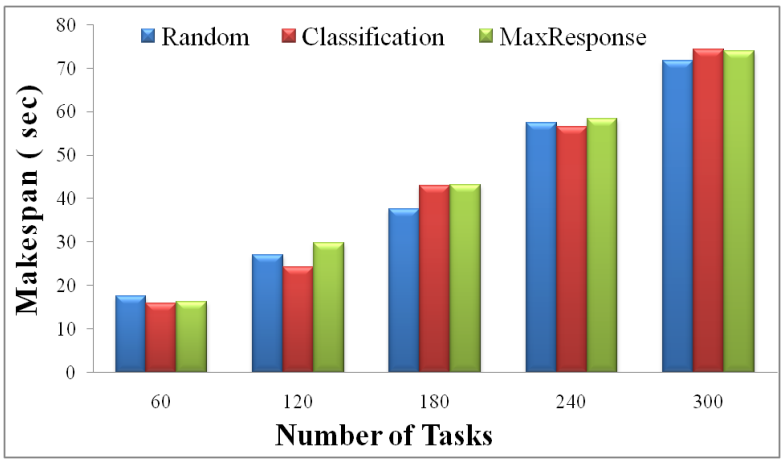
5.3.3 Varying Number of Tasks

Fig. 5.3 illustrates the impact of varying the number of tasks on the makespan, miss ratio and average delay metrics for the different task selection strategies. makespan, miss ratio and average delay grow at a steady pace with increasing the number of tasks. Applying different strategies has no significant

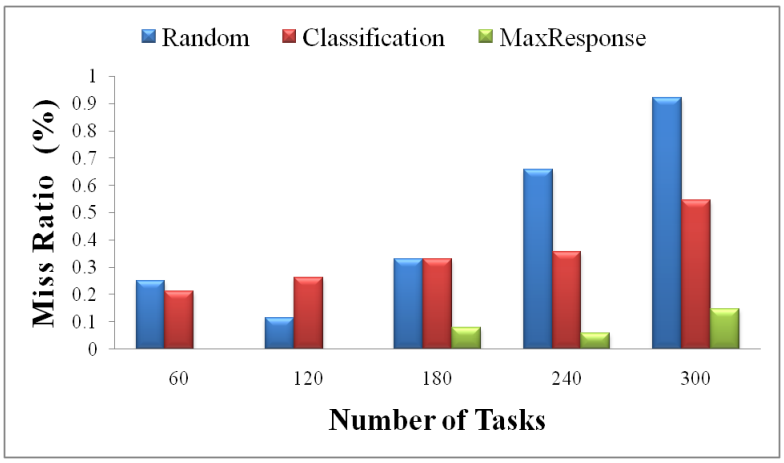
effect on makespan, as shown in Fig. 5.3(a). However, Figs. 5.3(b) and 5.3(c) show great differences on miss ratio and Avgdelay. Classification and MaxResponse strategies show less miss ratio and Avgdelay compared to Random strategy, since they qualify the choice of the candidate task to be reallocated. Also, MaxResponse strategy shows less miss ratio and average delay compared to the classification strategy since it reallocates the tasks that have maximum MaxResponse time with extra transmission time cost. Hence, the time-constrained tasks remain for processing with a better chance of finishing earlier. It is worth noting that the miss ratio when using the MaxResponse strategy is not exceeding 0.2% when the total number of tasks is 300. Also, the average delay does not exceed 5 seconds in the MaxResponse strategy, whereas it reaches about 25 seconds in the Random strategy.

5.3.4 Varying Number of Cloud Nodes

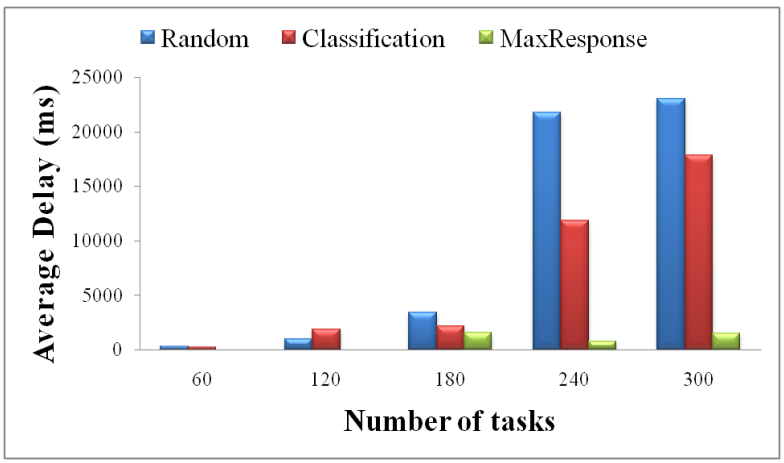
Fig. 5.4 demonstrates how varying the number of cloud devices affects makespan and the Cost with number of tasks is set to 60. It is worth noting that despite increasing the cloud resources raises the costs, no significant improvement in makespan is observed. The impact of varying the number of cloud devices when the number of tasks is 300 is shown in Fig. 5.5. Increasing the number of cloud devices results in increasing the cost and decreasing the makespan. So, based on our configurations and the capabilities of the fog devices, and the complexities of the tasks, it is not preferred to increase the cloud devices before the number of tasks reaches at least 300 tasks. This leads to monetary cost savings.



(a) Makespan

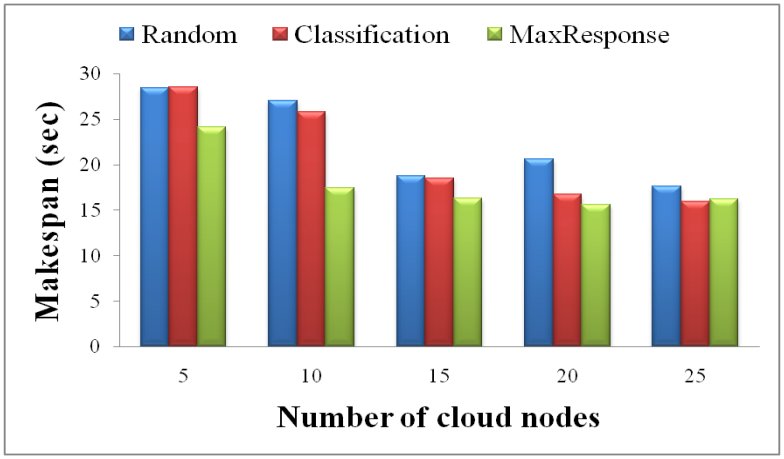


(b) Miss Ratio

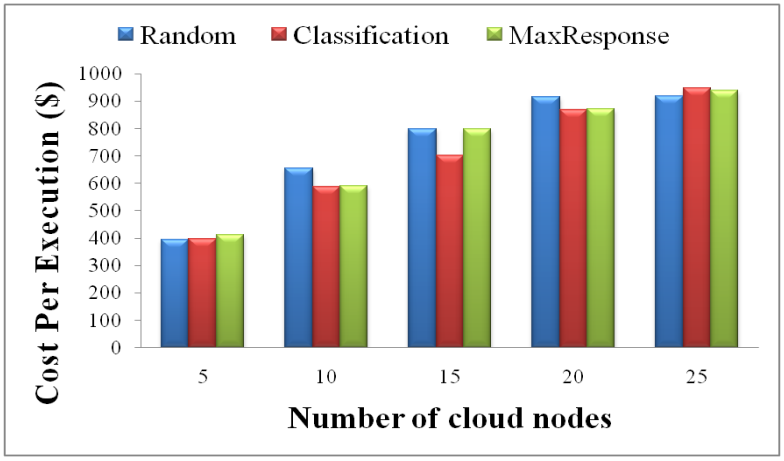


(c) Average Delay

Figure 5.3: Varying number of tasks

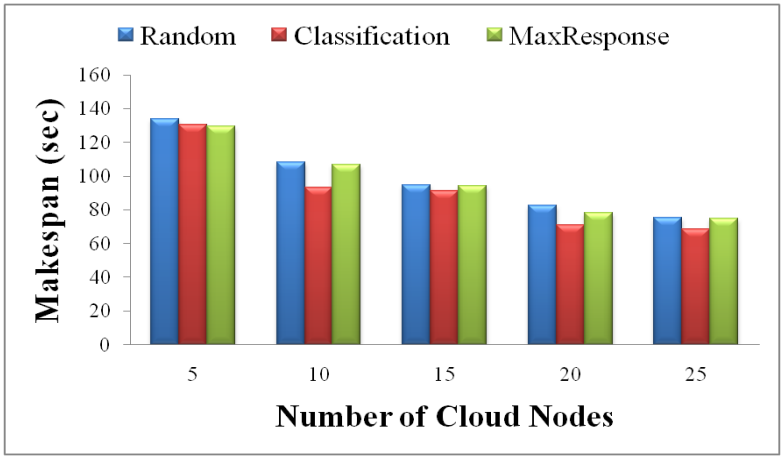


(a) Makespan

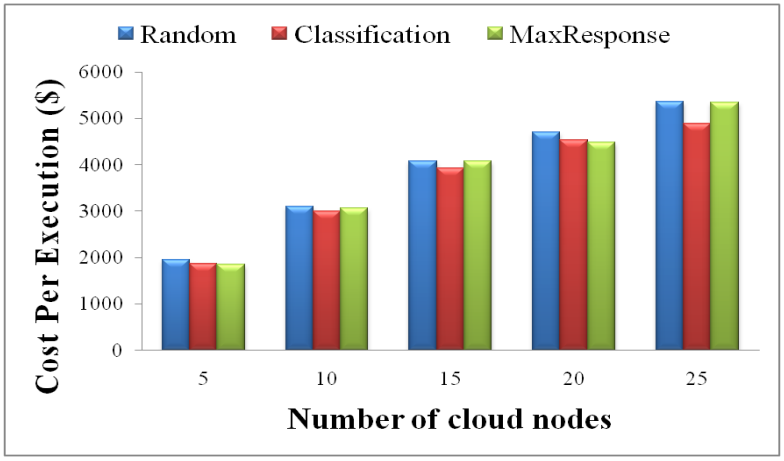


(b) Cost per execution

Figure 5.4: Varying number of cloud devices for 60 tasks



(a) Makespan



(b) Cost per execution

Figure 5.5: Varying number of cloud devices for 300 tasks

Chapter 6

Mobility-Aware Cloud-Fog based Architecture for IoT Healthcare

In this chapter, firstly, we propose a mobility-aware IoT healthcare architecture through the deployment and implementation of an RSS-based handoff mechanism, and a Mobility-aware Modified BAlance Reduce scheduling and allocation approach (MobMBAR). Secondly, we perform a comparative study against the state-of-the-art approaches. We compare against three against different paradigms and approaches in the vicinity of integrated Cloud-Fog IoT and Cloud-Only IoT. Thirdly, it includes a realistic simulation case study that uses the layout of an indoor hospital building in Chicago.

6.1 Proposed Mobility-Aware Task Scheduling and Allocation

Healthcare architectures encompasses patients who are human beings interacting and moving. While the patient is moving freely in the area, it is possible that he/she passes the wireless coverage range of the gateway device it is connected to and hence, being temporarily disconnected. This in turn results in network complexities in terms of the change in the location (spatial) and timing (temporal) of the data forwarding process, in addition to interruption in the services executed (healthcare tasks). Hence, this imposes a fundamental characteristic to be implemented in our architecture which is handling mobility through an appropriate handoff mechanism. This helps in the support of free movements of the patient without affecting the QoS that is represented in the accuracy, delay and cost in most cases. Consequently, we aim at allowing the mobility (physical activity) of the patients in the environment

through the deployment and implementation of a suitable handoff mechanism which is tightly related with the nature of our Cloud-Fog architecture. In addition, a mobility-aware task scheduling and allocation approach (MobMBAR) is proposed. We thereafter investigate more in the effect of the patients' mobility on our proposed scheduling and allocation approach (due to the approach being data locality based). We validate our proposed architecture and implemented algorithms in the case of immobility and mobility as well.

Referring to our proposed architecture in chapter 5, the moving patients are equipped with different body sensors (BIO sensors and activity sensors) which generate continuous raw data aggregated by mobile sink device such as a smart phone. This smart phone is connected to its upper layer fog device (gateway represented in access point or wireless router) through wireless communication (WLAN). Moreover, environmental sensors in IoT devices layer sense the information from the environment and are then collected by their connected static sink device such as micro-controller to be further forwarded to next layer (fog layer). Fog layer encompasses fog devices in addition to a specific device called fog broker representing the mastermind of the architecture. The cloud is the fourth and final layer in the architecture. Regarding the communication and networking technologies, we assume Bluetooth/wired connection between IoT devices layer (sensors) and the sink layer, WLAN between the sink layer and fog layer, LAN between fog devices in the fog layer, and WAN between the fog layer and the cloud layer.

This section firstly discusses the implementation details of the handoff mechanism utilized to support the movement of the mobile sink devices from a coverage range of a gateway device to another. This in turn adds more flexibility to the physical movement of the patient where the mobile sink device is attached to. The second subsection presents mobility-aware scheduling and allocation approach based on MobMBAR.

6.1.1 Handoff

Handoff in wireless networks refers to the transfer of connection between a mobile device and its gateway from a wireless network to another. Handoffs have been studied extensively and are usually classified into two categories: 1) horizontal handoffs (HHO), and 2) vertical handoffs (VHO). Ex-

amples on the handoff strategies in wireless networks both horizontal and vertical covering different wireless technologies such as IEEE 802.11 and LTE are [156], [157], [158], [159], [160], [161], [162], whereas handoff is explicitly studied for fog computing in [163]. In HHO, the mobile devices change their attachment between fog gateways in the networks of the same access technology. Whereas in VHO, the mobile device switches between fog gateways from different access technologies or network environments. In our architecture, we assume that all the sink devices communicate to the fog gateway through the same WLAN WiFi communication technology. Hence, we are following HHO implementation.

We have to note that the utilization of WiFi technology comes in view of the fact that our architecture covers small area such as indoor buildings e.g. a retirement home or hospital. Hence, short range technology is sufficient to fulfil our coverage requirement. On the contrary, long range technology covering remote ranges up tens of kilometers such as LTE-based cellular networks are intended to cover wider areas e.g. a small or medium sized city through distributed cellular operators. As a result of this privilege, LTE-based cellular networks causes a high drain in the power of the devices making use of them. This high drain is inadequate for battery constrained devices considered in the sink layer within our architecture. In addition to energy inefficiency, LTE-based cellular networks typically face two extra weaknesses which are chargeable services and lower data rate, despite recent attempts to catch up WiFi rates that is by some means successful. WiFi is preferred over other short-range networking technologies based on IEEE 802.15 such as ZigBee, 6lowPan, etc... because it is more powerful in terms of data rate, easily deployed and is able to avoid congestion.

We suggest CoAP over UDP protocols for application and transport layers respectively due to the constrained features of the sink devices. In addition, we think that UDP specifications best fits our architecture if we issue a trade-off between the re-transmission process for dropped packets which contains old vital data of the patient at the expense of delay, and between ignoring them and delivering the new health information on time.

We adopt the traditional HHO mechanism that uses as a basis the signal strength estimation [156]. Received signal strength (RSS) is typically a common metric for representing the quality of the wireless link. Almost the majority of the handoff mechanisms rely on this metric. RSS-based mechanism

is a lightweight, powerful which results in high handoff success rate. Sometimes, a cost function is built that utilizes other factors along with RSS such as bandwidth, power consumption, and velocity. However, multiple parameters can complicate the handoff algorithm which degrade the performance by adding more delay. In addition, the sink devices have limited capabilities to perform a complex handoff algorithm. In our scenario, our objective is to enable the mobile sink devices to gain a quick successful access to the network.

Our HHO mechanism is performed by defining two phases: 1) handoff decision policy, and 2) handoff strategy. In handoff decision policy phase, received signal strength (RSS) is utilized to give an indication that a moving device is being transferred from a wireless network area to another. We choose combining two conditions for confirming the decision of handoff. Hence, a handoff decision is taken if

$$(RSS_{GW} < RSS_{Th}) \wedge (RSS_{new} > RSS_{GW}) \tag{6.1}$$

is true where RSS_{GW} is the received signal strength from the fog gateway at the mobile sink device, RSS_{TH} is the received signal strength threshold, and RSS_{new} is the received signal strength from the closest new fog gateway. This decision policy results in lower number of handoff failures, and hence less number of missed tasks. We can control the number of handoff operations by adjusting RSS_{TH}. Smaller number of handoff operations results in reduction in latency cost. Hence, it is important to appropriately set RSS_{TH} value that keeps the number of handoff operations reasonable. For the choice of RSS_{TH}, we need to choose a threshold value equal to or smaller than -70 dBm which is approximately the minimum value for reliable data delivery in WiFi and cellular systems [164], [165].

We also rely on RSS as a handoff strategy. In this strategy, when a handoff decision is taken, a sink device chooses the closest fog gateway device to connect to while we take into consideration the maximum number of devices that can be connected to the fog gateway [157], [166] and [167]. The complete handoff procedure is shown in figure 6.1.

We follow log-distance path loss model for indoor WiFi in Eq.6.2 [168].

$$\log d = \frac{1}{10n}(P_{Tx} - P_{Rx} + G_{Tx} + G_{Rx} - X_{\sigma} + 20\log\lambda - 20log(4\pi)) \tag{6.2}$$

d marks the distance between the mobile sink device and the candidate fog gateway, P_{Tx}, P_{Rx} are the radio transmit and receive power respectively, G_{Tx}, G_{Rx} are the antennae gain of the transmitter and

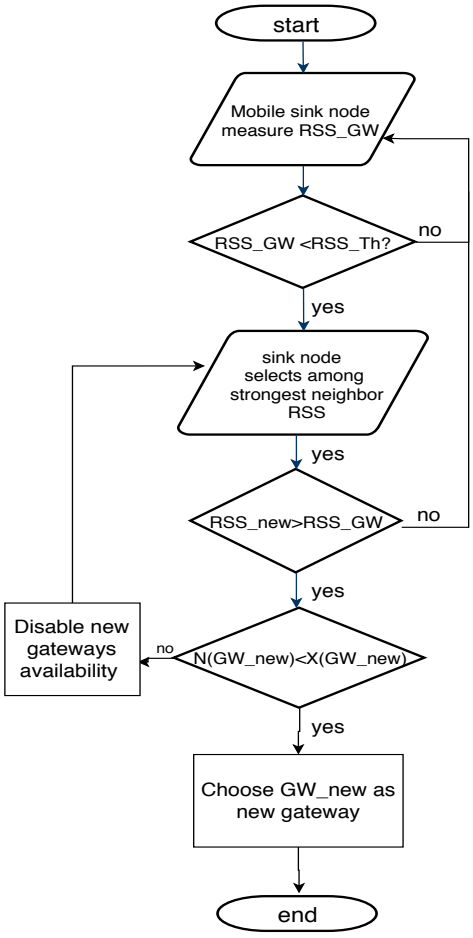


Figure 6.1: Proposed RSS-based handoff mechanism

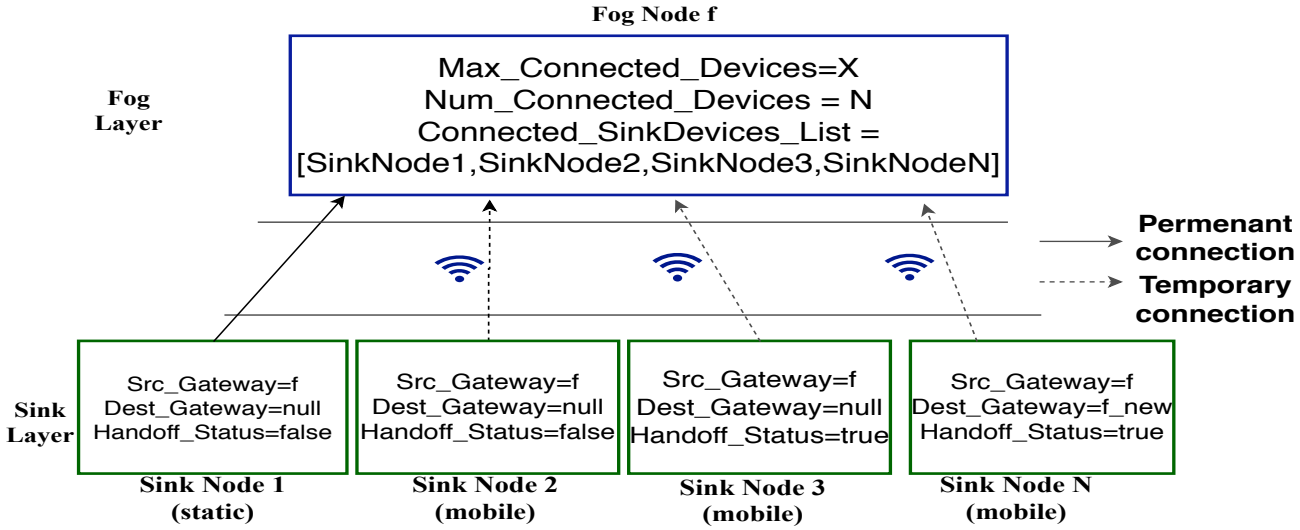


Figure 6.2: Communication between Sink Layer and Fog Layer

receiver respectively, $\lambda(m)$ is the wavelength of the signal, n is the propagation constant, and X_σ is a normal random variable with a standard deviation equal to σ. Note that after the handoff strategy is performed, the members list of the fog gateway is updated. This list is continuously reported to the fog broker through *resource collector* module and defined as data placement information. Figure 6.2 illustrates the connection between the sink layer and the fog layer in our architecture and how the handoff is handled. Figure 6.2 describes a sample case for a fog device called f connected to four sink devices: 1 static sink device (through LAN or WLAN), and 3 mobile sink devices (through WLAN). The static device is micro-controller that is responsible for the data collection from the environmental sensors. The mobile sink devices represents smart phones, smart watch, or any easy body held embedded device that is able to gather the data from the body sensors. Data collection from the sensors at the sink devices is performed through wired or Bluetooth communication. The figure shows 4 scenarios for the handoff process at time t. All of the 4 sink devices has their source fog gateway f.

- The first device $SinkNode1$ is static and hence, is always connected to the same fog device f because of its immobility. Its destination fog gateway is always equal to $null$ and its handoff status is always equal to $false$.

  The second, third and fourth sink devices are mobile and represent different cases at time t.

- $SinkNode2$ is a mobile sink device, at time t, it is still in the safe transmission range of f.
- $SinkNode3$ encounters a handoff decision so its handoff status is set to $true$ which means that the RSS_f value is less than RSS_{Th} while still searching for the destination fog gateway.
- $SinkNode4$ also encounters a handoff under the same condition as $SinkNode3$, and started it handoff strategy to choose the destination fog gateway which is chosen to be f_{new}.

The attributes of the f describe: the most extreme devices number able to be associated with it which depends on the type of the device in addition to the communication technology (X), the actual number of devices (N) connected at time t, and the members list holding objects representing the sink devices connected.

6.1.2 Mobility-Aware Modified BAlance-Reduced (MobMBAR) algorithm

As described in chapter 5, our proposed scheduling and allocation approach consists of two phases: 1) Ranking phase, and 2) Scheduling and allocation phase. The first phase adapts and implements a multi-criteria decision making algorithm called Weighted Sum Method (WSM) for ranking the tasks and is explained in section 5.2.2. However, in this section, we upgrade and extend the second phase of the proposed scheme to support the mobility as shown in Algorithm 2. MobMBAR adapts the scheduling and allocation of the health tasks according to the movement of the patients and hence, their spatial and temporal data placement. MobMBAR algorithm works on minimizing the makespan described in Eq. 5.9. In **balanced step**, MobMBAR follows procedure explained in 5.2.3, however, if task t_i is found to be attached to a sink device that has performed a recent handoff process, The transmission time is calculated as:

$$t(t_i, P) = D_{t_i(partial)}/BW_P \tag{6.3}$$

where $D_{t_i(partial)}$ is the size of partial measurement data. The reported raw data is partially located at the source fog gateway and partially located at the destination fog gateway or fully located at source or destination fog gateway based on the time the handoff process is performed. For example, if the handoff is performed at the start of the interval α, all of the task data will be at the destination fog

gateway. And if the handoff is performed at the end of the interval α, all of the task data will be at the source fog gateway. Consequently, $D_{t_i(partial)}$ at clock CLK is calculated according to Eq. 6.4.

$$D_{t_i(partial)} = \begin{cases} [Clk_{HO} - (Clk - \alpha)] \times \dfrac{D_{t_i}}{\alpha}, & \text{if } P = GW_{Dest} \\ [Clk - Clk_{HO}] \times \dfrac{D_{t_i}}{\alpha}, & \text{if } P = GW_{Src} \end{cases} \tag{6.4}$$

where D_{t_i} is the volume of collected data, Clk is the current clock when the allocation is executed and Clk_{HO} refers to the clock at which the handoff started. GW_{Src} is the preceding fog gateway device connected to the sink mobile device that executes t_i. GW_{Dest} is the current fog gateway device connected to the sink mobile device that executes t_i.

In **Reduce step**, we choose task with higher Class on P_{max} to reallocate it on a remote computing device according to Eq. 6.5.

$$t_{reallocate} = \max_{t \in \beta(P_{max})} Class_t \tag{6.5}$$

Algorithm 2 Pseudocode for MobMBAR

Define $\mathcal{U}$ as a set that describes input tasks
Define $\mathcal{V}$ as a set that describes fog devices
Define $\mathcal{W}$ as a set that describes cloud devices
Define $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ as bipartite graph of sets $\mathcal{U}, \mathcal{V}$
Define (t_i, d_i) as $(task, IoTdevice)$ pair
Define $List_{handoffs} = \{d1, d2, ...\}$ as a list of devices that performed succesful handoff since last allocation $\beta_{CLK-\alpha}$

Phase 1 – Local allocation

procedure BALANCE
Input: Bipartite graph $\mathcal{G}$
Output: Allocation β
  for each task t_i (represented by vertex $v1 \in \mathcal{U}$) **do**
    for each processing node P (represented by vertex $v2 \in \mathcal{V}$ such that $\exists\, e(v1, v2) \in \mathcal{E}$) **do**
      if Inlist(d_i , $List_{handoffs}$) **then**
        $L_P(t_i) \leftarrow CalcWorkload(D_{ti(partial)}, \theta_{ti})$
      else
        $L_P(t_i) \leftarrow CalcWorkload(\theta_{ti})$

Algorithm 2 Pseudocode for MobMBAR (continue)

```
            end if
        end for
        β(t_i) ← MinWorkload{L_P(t_i)}
        L_P^fin ← L_P^fin + L_P(t_i)
    end for
end procedure
```

Phase 1 – Remote allocation

```
procedure REDUCE
Input: Bipartite graph 𝒢, 𝒲, β
Output: Allocation β, Makespan
    P_max ← MaxWorkload{L_P^fin}
    Makespan_old ← L_Pmax^fin
    β_old ← β
    do
        t_reallocate ← MaxClass{β(P_max)}
        Makespan_expected ← L_Pmax^fin − L_Pmax(t_reallocate)
        L_Pmax^fin ← L_Pmax^fin − L_Pmax(t_reallocate)
        β ← β − β(P_max)
        for each P ∈ 𝒱, 𝒲 such that P ≠ P_max do
            L_P(t_reallocate) ← CalcWorkload(D_t_reallocate, θ_t_reallocate)
        end for
        β(t_reallocate) ← MinWorkload{L_P(t_reallocate)}
        L_P^fin ← L_P^initial + L_P(t_reallocate)
        P_max ← MaxWorkload{L_P^fin}
        Makespan ← L_Pmax^fin
        if Makespan ≤ Makespan_expected then
            β_old ← β
            Break
        else if Makespan_expected < Makespan && Makespan < Makespan_old then
            Return
        else
            β ← β_old
            Makespan ← Makespan_old
            Return
        end if
    while
end procedure
```

6.1.3 Energy Analysis Model to Proposed Approach

In this section, we analyze to network and computing energy consumption of IoT-Fog-Cloud scheduling and allocation approach. We target the fog and sink devices for being constrained IoT devices. We focus on the network and computing energy consumed for the whole allocation process to be carried out. We firstly model the energy consumption for computations on the fog devices. Next, we model the energy consumption for the networking aspects.

Computing Energy

We denote the energy consumed by the CPU of fog device as E_f^{comp}. This parameter equals the combination of the static and dynamic energy consumed [169], [103]. The static energy is the energy consumed during the idle state of the CPU. The dynamic energy is the energy consumed during the execution of the tasks allocated to that fog device .

$$E_f^{comp} = E_f^{comp-static} + E_f^{comp-dynamic} \tag{6.6}$$

such that

$$E_f^{comp-static} = P_f^{IdleCPU} \times (Makespan - L_f^{fin}) \tag{6.7}$$

and

$$E_f^{comp-dynamic} = P_f^{ExecCPU} \times L_f^{fin} \tag{6.8}$$

Each fog device exerts time L_f^{fin} to process all the tasks allocated to it. This value is equal to the final workload assigned by the allocation algorithm. Otherwise, the fog device is considered idle. $P_f^{IdleCPU}$ and $P_f^{ExecCPU}$ refer to the power consumption of the CPU at idle state and processing state respectively. These values depend on the energy profile of the fog devices [169]. Consequently, the total computing energy consumed by all the fog devices can then be calculated as:

$$E_F^{Total-comp} = \sum_{j=1}^{\mathcal{V}} [P_j^{IdleCPU} \times (Makespan - L_j^{fin}) + P_j^{ExecCPU} \times L_j^{fin}] \tag{6.9}$$

Networking Energy

E_f^{comm} refers to the energy consumed by fog devices communications.

$$E_f^{comm} = E_f^{comm-static} + E_f^{comm-dynamic} \tag{6.10}$$

such that

$$E_f^{comm-static} = P_f^{IdleNIC} \times (Makespan - T_f^{comm-recv}) \tag{6.11}$$

and

$$E_f^{comm-dynamic} = P_f^{RecvNIC} \times T_f^{comm-recv} \tag{6.12}$$

$E_f^{comm-static}$ can be calculated by multiplying the power consumed by the NIC at the idle state times the duration when the fog device is in the networking idle state (not receiving data). Correspondingly, $E_f^{comm-dynamic}$ is calculated by multiplying the power consumed by the NIC at the receive mode, times the duration the fog device is receiving data needed to process the task if needed. $T_f^{comm-recv}$ refers to the time the fog devices consume in receiving the data which is equal to the total size of transmitted data for remote processing divided by the bandwidth of the transmission links and is hsown in equation 6.16. It is proportional to the number of tasks partially locally and remotely allocated based on the stationary or mobility status of the sink devices generating the task. Hence, equations 6.14 and 6.15 refer to the time the fog devices consume in receiving the partially and remotely allocated tasks respectively.

$$T_f^{comm-recv} = T_f^{comm-recv-stat} + T_f^{comm-recv-mob} \tag{6.13}$$

Stationary scenario:

$$T_f^{comm-recv-stat} = \frac{\sum_{i=1}^{N} D_{t_i} \delta(\beta(t_i) - f) \delta(\mathcal{B}_{\mathcal{CLK}}(t_i, f))}{BW_f} \tag{6.14}$$

Mobility scenario:

$$T_f^{comm-recv-mob} = \frac{\sum_{i=1}^{N} D_{t_i(partial)} \delta(\beta(t_i) - f) \mathcal{B}_{\mathcal{CLK}}(t_i, f) \delta(\mathcal{B}_{\mathcal{CLK}-\alpha}(t_i, f))}{BW_f} \tag{6.15}$$

$\delta(\beta(t_i) - f)$ is the kronecker delta function between the allocation of task t_i denoted as $\beta(t_i)$ and fog device f such that it is true if they are equal and false otherwise. $\mathcal{B}$ is the biadjacency matrix of bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$. $\mathcal{B}(t_i, f)$ is true if there exist a data locality between task t_i and fog device f. Hence, in stationary scenario, two conditions have to be met to consider the transmission of data needed for task t_i.

1. The first condition is that task t_i is allocated to fog device f ($\beta(t_i) = f$).
2. the second condition becomes the non-existence of data locality between task t_i and fog device f.

On the other hand, in mobility scenario, an energy is consumed for transmitting the partial data resulting from the handoff process. In this case, three conditions have to be met to consider the transmission of partial data needed for task t_i.

1. The first condition is that task t_i is allocated to fog device f ($\beta(t_i) = f$).
2. The second condition is the existence of data locality between task t_i and fog device f in the current allocation.
3. The third condition the non-existence of locality between task t_i and fog device f in the previous allocation $\mathcal{B}_{\mathcal{CLK}-\alpha}$.

Consequently, the total networking energy consumed by all the fog devices can then be calculated as:

$$E_F^{Total-comm} = \sum_{j=1}^{\mathcal{V}} [P_j^{IdleNIC} \times (Makespan - T_j^{comm-recv}) + P_j^{RecvNIC} \times T_j^{comm-recv}] \tag{6.16}$$

Regarding the sink devices, the energy consumed to transmit the data aggregated by sink devices to their attached fog gateway devices is calculated as:

$$E_s^{comm} = P_s^{TranNIC} \times \frac{\sum_{i=1}^{N} D_{t_i}}{BW_s} \tag{6.17}$$

The total energy consumed by the sink devices is equal to the power consumed by NIC when it is at the transmit mode, multiplied by the total time needed to transmit all the data from the sink devices to their attached fog gateway devices. We assume that the sink device is awake only during the transmission of sensed data and sleep otherwise. As an example, based on our utilization of the WiFi technology, $P_s^{TranNIC}$ can be calculated based on the values reported in [170]

6.1.4 Performance Evaluation

This section presents an evaluation of the performance for the approach proposed after its updated to support mobility. We carry out our simulations utilizing iFogSim [155]. As reported in chapter 5, iFogSim models a complete environment of fog computing and is built upon CloudSim which is a well-known simulator for simulating cloud computing. However, we had to extend the simulator to support mobility for randomly selected devices. The environment and simulation setup is firstly described followed by an enumeration of the configurations for the devices. For fair comparison, we tune their parameter to the exact values of the work in [110] regarding the number and capabilities of fog, cloud devices, and the number of tasks.

The fog environment area is set to 700×700 meter where 50 fog devices are uniformly distributed over the area at interval of 100 meters as shown in figure 6.3. The environment encompasses 50 sink devices. The locations of sink devices are randomly generated in each scenario. The mobile sink devices (smart phone hanged to the patient) are following random waypoint mobility model with no pause time, and with a fixed speed at 1.4 m/s which is human average walking speed [171], [172], and [173]. We assume that the sink devices communicate to the fog gateway through Wi-Fi technology. Hence, we apply indoor WiFi path loss model shown in Eq. 6.2 by setting the parameters of the path loss model as shown in table 6.1. We set n = 3.8 assuming an indoor building environment with obstacles. WLAN transmission range reaches up to almost 75 meters. The maximum number of connected sink devices to access point is set to 50. Table 6.2 shows the simulation setup where the duration of simulation is 30 minutes and the tasks are periodically allocated every 10 seconds ($\alpha = 10$), this sums up to 180 total allocation runs. The fog environment is connected to cloud datacenter consists of 60 cloud devices. The cloud devices are 200 times the fog devices in the computing capability.

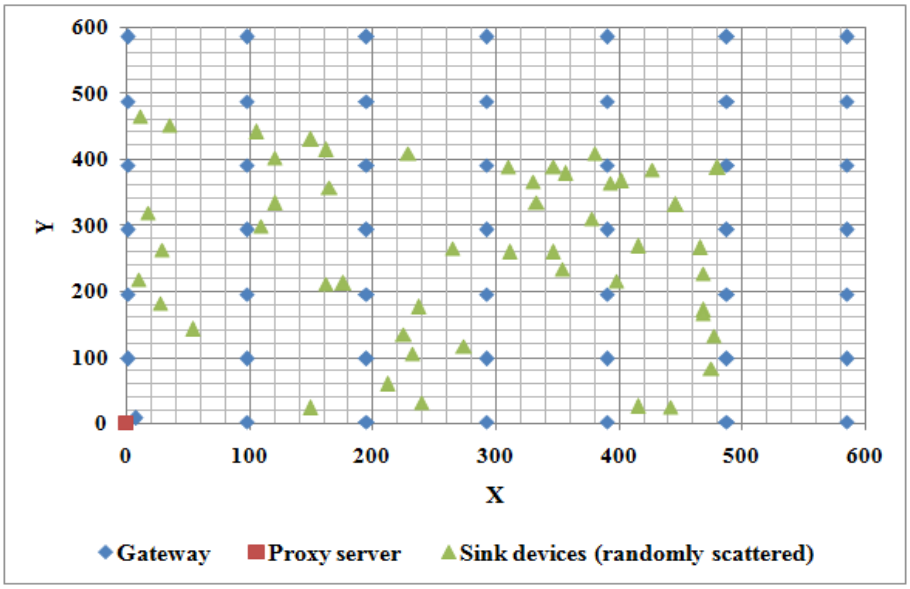


Figure 6.3: Environment setup

Table 6.1: Path-loss model parameters

Parameter	**Value**
P_{Tx}	100 mW
G_{Tx}	2.5 dBm
G_{Rx}	2.5 dBm
$\lambda(m)$	0.12m
σ	3 dBm
n	3.8

The details of the cloud, fog, and sink devices configuration are described in tables 6.3, 6.4, and 6.5 respectively. The corresponding energy related values for fog devices and Ethernet NIC from [103], and the corresponding energy values for IEEE 802.11n NIC from [174] and [175] are listed. Configuration details of the tasks features are shown in table 6.6 which describes the configuration details we implemented.

Table 6.2: Simulation Setup

Parameter	**Value**
Duration	30 minutes
α	10 seconds
Speed of mobile sink	1.4 m/s
Gateway(Access Point)	transmission range: 75 meters, maximum connected sinks: 50

Table 6.3: Cloud devices parameters

Parameter	**Value**
Cloud devices number	60
Processing capacity	2250 MIPS (2.2 GHz)
RAM	32 GB
Storage	11 TB
Bandwidth	100 Mbps
Cost per time unit	3 $

Evaluation Metrics

In addition to makespan described in 6.1.4 and specified using Eq. 5.9, we extend the evaluation metrics to include more analysis and to compare with state-of-the-art. We evaluate the metrics: Network load, number of handoffs, missed tasks, and energy consumption E_{total}. Network load represents total size of data transmitted on the network links. Number of handoffs is the total number of handoff operations performed during the simulation duration. Missed tasks is the percentage of tasks that are not processed due to failed handoff process by its attached sink device. E_{total} computes the total energy consumed as discussed in subsection 6.1.3 and is equal to the summation of equations 6.9, 6.16 and 6.17.

$$NetworkLoad = \sum_{\beta(t_i) \in \mathcal{W}} D_{t_i} \tag{6.18}$$

Table 6.4: Fog devices parameters

Parameter	**Value**
Fog devices number	50
Processing capacity	11 MIPS
RAM	8 KB
Storage	92 KB
Bandwidth	1024 Mbps
Idle CPU power consumption	0.36 Watt
Execution CPU power consumption	0.44 Watt
Idle NIC power consumption	10^{-14} Watt
Receive NIC power consumption	0.285 Watt

Table 6.5: Sink devices parameters

Parameter	**Value**
bandwidth	300 Mbps
Transmit NIC power consumption	1.254Watt

Table 6.6: Tasks parameters

Parameter	**Value**
Complexity	[1000, 20000] MI
Data size	[100, 500] MB
Max response time	[15 sec, 30 min]

$$MissedTasks = \frac{No.\,of\ non-processed\,tasks}{Total\,issued\,tasks} \tag{6.19}$$

$$E^{Total} = E_F^{Total-comp} + E_F^{Total-comm} + E_s^{comm} \tag{6.20}$$

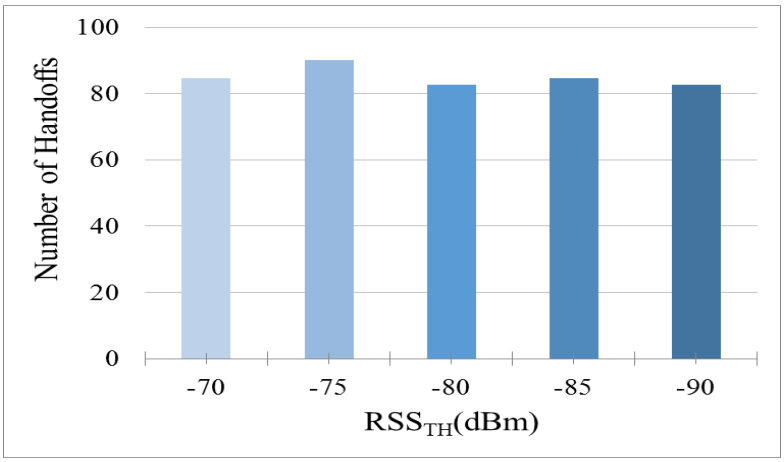
All result are averaged over 10 runs which means 10 different scenarios for each evaluation criteria. The variations are: 1) The class and configuration of tasks, 2) The direction of the moving sink device: {south, north, east, west, south-east, south-west, north-east, north-west}, and 3) x,y coordinates for initial location of moving device. All the variations are maintained within the described ranges in tables 5.4 and 6.6. Diversity in scenarios is intended to provide the assessment under various circumstances to best evaluate the scalability and generality of the proposed architecture.

Varying Hand-off Threshold

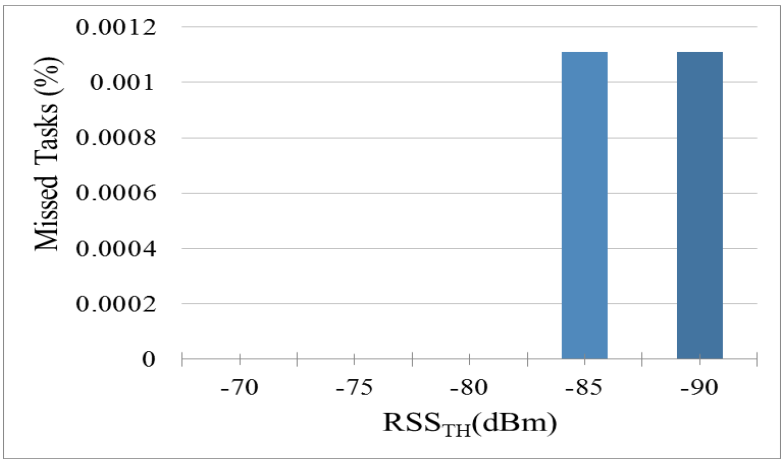
Figure 6.4 illustrates the effect of varying received signal strength threshold (RSS_{Th}) on the handoffs as well as the missed tasks. The range of RSS_{TH} is -70 dBm to -90 dBm which is the sensitive value we consider for disconnection. The percentage of mobile devices is 40% and the number of sink devices is 50. It can be noticed that the number of handoff operations reaches a maximum value of 90. It is also observed that the number of handoff operations is very close for different thresholds. The number of missed tasks is zero when RSS_{Th} equal to -70dBm, -75dBm and -80dBm, and reaches 0.001% as a maximum value when RSS_{Th} is equal to -85dBm and -90dBm. It is worth noting that changing RSS_{Th} has no significant effect on handoff operations number nor missed tasks number. This is due to that our proposed handoff scheme considers also the strength of signal of the fog gateway that the sink device is moving to.

Varying Percentage of Mobile Sink Devices

Figure 6.5 shows the result of varying the percentage of mobile devices ranging from 0% to 100% when the number of sink devices is 50. Figure 6.5(a) shows close values for makespan ranging from 44 to 47 seconds. The insignificant variations between the different percentages is noticed due to the random waypoint mobility factors for the mobile sink devices (direction and initial location). It must be taken into consideration that when the sink devices start to move and change their fog gateway as a result of the handoff process, the workload cost is affected. This results from the necessary additional time required to transmit part of the data to new/old fog gateway even if the task is processed locally. However, MobMBAR approach is still able to sustain the stability of makespan values which means

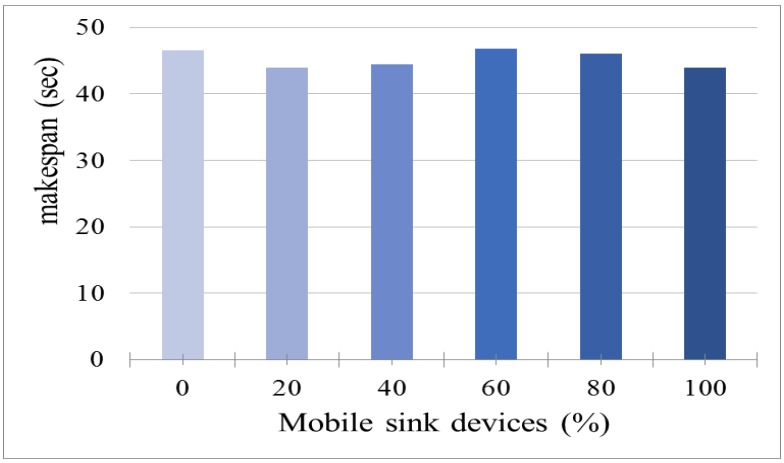


(a) Number of handoffs

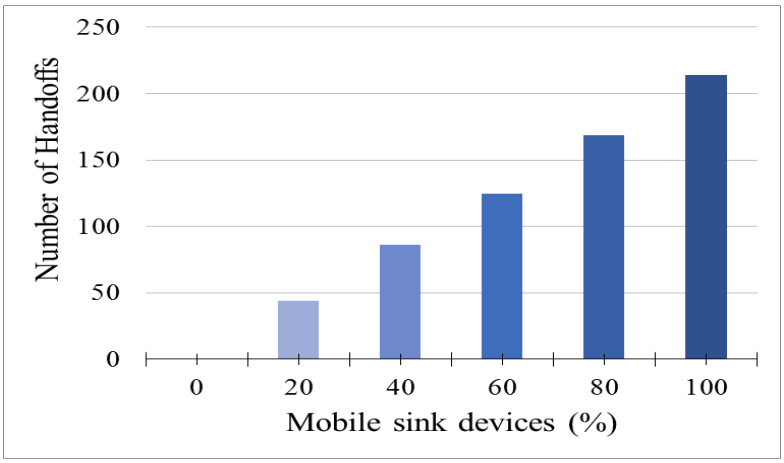


(b) Percentage of missed tasks

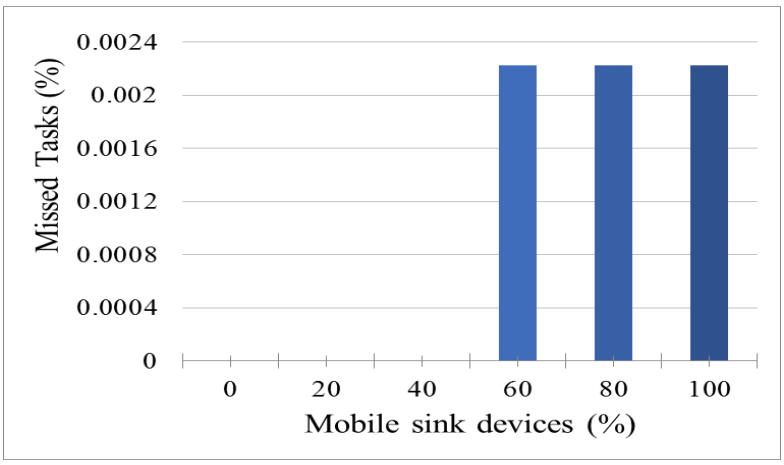
Figure 6.4: Varying handoff threshold



(a) Makespan



(b) Number of handoffs



(c) Percentage of missed tasks

Figure 6.5: Varying percentage of mobile devices

Table 6.7: Resources utilization

(a) Resources utilization

Device	**Allocation Percentage (out of 27000 task)**	**Classification of the tasks allocated**		
		Data analysis	**Context Management**	**Critical Analysis /Critical Control**
Cloud	54.7%	59%	33.7%	7.3%
Fog	45.3%	5%	30%	65%

(b) Task Allocation

Task Classification	**Allocation Percentage on devices**	
	Cloud	**Fog**
Data analysis	93.5%	6.5%
Context management	57.5%	42.5%
Critical Analysis / Critical Control	12%	88%

close values with time difference less than 3 seconds for different percentage of mobile sink devices. This proves the scalability of the approach. Figure 6.5(b) shows that the number of handoff operations increases when the percentage of mobile devices is increased. Figure 6.5(c) demonstrates that the number of missed tasks is equal to zero for percentages of mobile sink devices equal to 0%, 20%, and 40%. Starting from percentage of mobile sink devices equal to 60%, the percentage of missed tasks is increased to be around 0.002%.

Resources Utilization

Table 6.7 show the result for the fog and cloud resources utilization for allocating 150 tasks with 60% of the sink devices are mobile, and makespan is equal to 46.8 seconds deducted from figure 6.5(a). We have to note that the allocation is executed 180 times during 30 minutes summing up to 27000 total tasks. It can be seen from table 6.7(a) that the allocation of the tasks is shared among the gateway fog devices and the cloud devices. Amongst the tasks allocated on the cloud devices, 59% are classified as data analysis, 33.7% as context management, and 7.3% as critical tasks. On the contrary, 65% of the

tasks allocated on the fog devices are classified as critical tasks, 30% as context management, and 5% as data analysis tasks which constitutes the least percentage. From table 6.7(b), it is also observable that high percentage of High-Level class tasks (critical analysis and critical control) are allocated on the fog devices 88%. While high percentage of Medium-Level class tasks (context management) are allocated on the cloud devices 57.5%. Also, high percentage of Low-Level class tasks (data analysis) are allocated on the cloud devices 93.5%. This comes from the fact that the higher the level of a task, the earlier it is ordered in the tasks queue (due to the ranking phase). Next, the leading tasks in the queue to be allocated are chosen to be processed on their local processing fog devices such that their time constraints are met.

6.2 Comparative Study

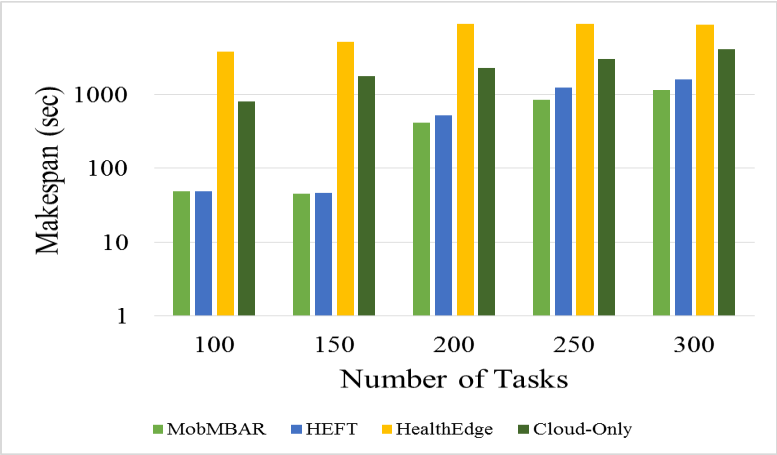
This section compares the performance of the proposed architecture against different architectures and approaches in the vicinity of integrated Cloud-Fog IoT and Cloud-Only IoT. These comparisons are performed under different scenarios and with detailed description of the varying parameters for the healthcare applications. We compare our work with: HealthEdge, HEFT approaches, and *Cloud-Only* architecture.

Firstly, we compare MobMBAR with HealthEdge, the single existing approach in literature for healthcare Cloud-Fog IoT architecture [110]. The task scheduling algorithm implemented in HealthEdge targets minimizing the schedule length and the network load. HealthEdge is based on two factors for determining the priority of the task which are the emergency factor and the data size of the task. However, in MobMBAR, priority determination relies on classification and maximum response time of the task. HealthEdge statically links each fog device with five tasks without taking into consideration the mobility of the patients. Whereas our proposed approach handles this mobility through the implementation of MobMBAR. Secondly, we compare MobMBAR with HEFT [176]. HEFT is a well-known heuristic algorithm for scheduling which seeks for the minimization of the makespan for the scheduled tasks. HEFT is adopted as an algorithm in Cloud-Fog IoT architecture proposed in [72]. While the authors in [72] propose a makespan- monetary cost trade-off for tasks scheduling on cloud and fog computing devices, our approach targets the latency. Due to the time criticality nature of the health-

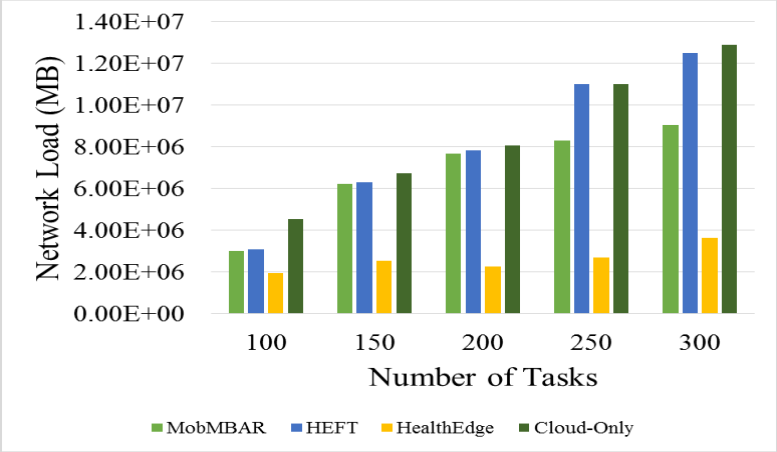
care tasks. However, we build directed acyclic graph (DAG) with no hierarchy (precedence constraint between two tasks is set to zero). Finally, we compare against *Cloud-Only* architecture. In this implementation, all the tasks are sent to be processed on the cloud devices. We choose to schedule the tasks using the *first come, first served (FCFS)* scheduling algorithm. This is achieved by performing a rotation on the cloud devices in consecutive order and then allocating the next task in the ready queue.

Figure 6.6 demonstrates the influence of varying the number of tasks on the makespan, network load in addition to energy consumption in case that 50% of the sink devices are mobile for a simulation duration of 30 minutes leading to 180 total consecutive allocations. We set the y axis to logarithmic scale in figures Figure 6.6(a) and Figure 6.6(c) to be able to notice the distinction between the four approaches.

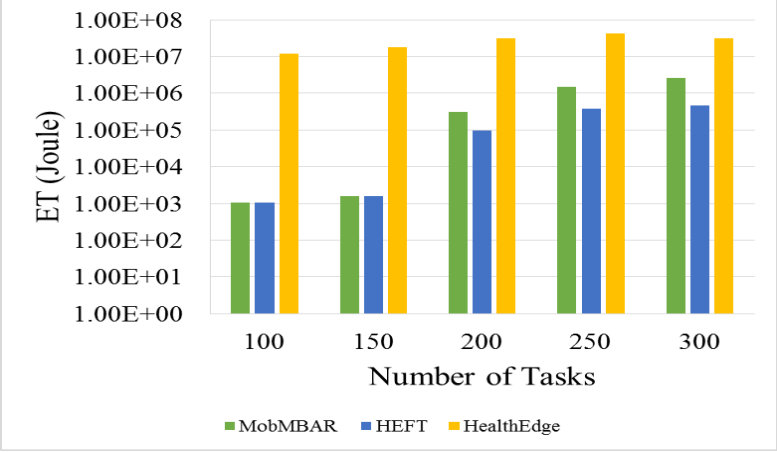
Figure 6.6(a) shows that MobMBAR outperforms the other approaches. It is observed that the makespan in HealthEdge exceeds 16 minutes (1000 seconds), when the number of tasks is 100 while it reaches only 48 seconds in MobMBAR and HEFT, and about 800 seconds (13 minutes) for *Cloud-Only*. Increasing the number of tasks causes an increase in the makespan for all the approaches. Despite that MobMBAR and HEFT show similar values for makespan when the number of tasks is 100 and 150, starting from 200 tasks, HEFT presents higher makespan This result of MobMBAR performing better than the other approaches is achieved due to multiple reasons. Firstly, when comparing to *Cloud-Only* architecture, MobMBAR makes use of fog devices beside the cloud devices as processing devices when performing the allocation. This leads to saving in extra transmission costs to the remote cloud datacenters. Secondly, MobMBAR takes into consideration the locality of data that resides in the gateway devices. This limits the extra cost caused by the transmission time to remote processing devices. However, HEFT and *Healthedge* do not consider data locality in scheduling decision. It is noticed that HealthEdge demonstrate the worst makespan because of its dual objective optimization function which tends to minimize the network traffic load at the expense of the makespan. In addition, HealthEdge chooses to allocate the tasks locally on the fog device (throught placement in *edge-queue*) when their emergency level exceeds certain threshold regardless of the capabilities of the fog device nor the requirements of the task to be processed. Note that we set the y axis to logarithmic scale to be able to notice the distinction between the four approaches. We have to point out that MobMBAR has



(a) Makespan



(b) Network load



(c) Energy consumption

Figure 6.6: Varying number of tasks

makespan values that is always below the $MaxResponse$ time for all the tasks (1800 seconds).

Figure 6.6(b) shows the network traffic load versus the number of tasks for the four approaches. It can be shown that HealthEdge has the least value for network load due its dual objective optimization function. However, the network traffic load in MobMBAR is lower than both HEFT and *Cloud-Only* approaches for different number of tasks particularly when the number of tasks is increased. Despite we can notice similar values when we compare HEFT with *Cloud-Only*, *Cloud-Only* witnesses higher network load due to its in-dependability on auxiliary processing devices (fog devices) beside the cloud devices.

Figure 6.6(c) displays the energy consumption of the fog and sink devices against the number of tasks for MobMBAR, HEFT, and HealthEdge. Figure 6.6(c) shows close energy consumption values for MobMBAR and HEFT at 100 and 200 tasks. When increasing number of tasks above these values, MobMBAR shows higher value due to its objective for decreasing the latency of the makespan and thus utilizing more fog devices. While HealthEdge tries to minimize the schedule length and traffic load, it may allocate more tasks to fog devices in order to reduce the network load for transmitting to the cloud. This may cause reduction in networking energy, but the computing energy will be greatly increased based on the nature that fog devices have a limited processing capabilities compared to cloud devices. An increase is noticed for all the three approaches when the number of tasks is increased.

6.3 Case Study: Simulations in Real Condition

This section presents a case study to evaluate proposed scheduling and allocation approach MobMBAR on an indoor hospital building environment in [177] with layout shown in figure 6.7. It consists of 5 patient's rooms and a nurse station with approximately $33\ m^2$ each. Each room is occupied with number of patients varying between 1 and 6. Each patient is attached to a sink device that collects data from 2 BIO sensors. In addition, 3 static sink devices collect data from 14 environmental sensors that are deployed in the building. Patient's movement follows constrained mobility (CM) model [178], which is a graph-based mobility model targets indoor environment with obstacles. CM model represents source and destination points by the graph vertices, and possible paths by graph edges. Movements of the patients between vertices should follow shortest path algorithm. Regarding fog

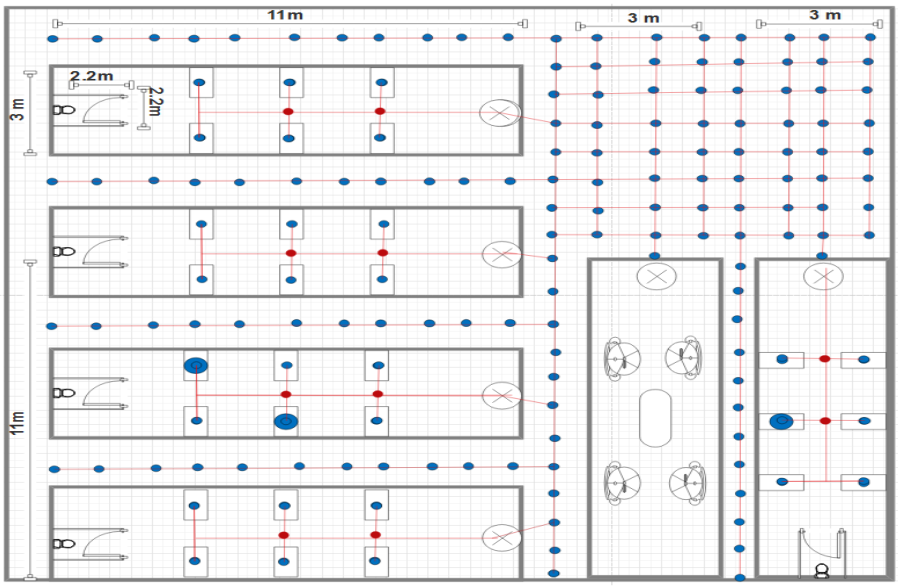


Figure 6.7: Hospital building layout

layer, Access Points AP are served as fog devices. Following Cisco estimation strategy [179] which sets the maximum of the number of APs based on throughput and client count, results in 4 APs for full hospital occupancy. Intel Edison Board with CPU of 500 MHz, 1GB RAM, 4GB storage, and Bluetooth and WiFi, is selected as AP with limited capability. The board consumes 0.385Watt at CPU idle state with WiFi enabled, 0.435Watt at CPU execution, NIC idle power consumption is 10-14Watt, and NIC receiving power consumption is 0.285Watt [180]. For cloud layer, we simulate one cloud device with 2.2GHz CPU, 32GB RAM, 11TBStorage, and 100Mbps bandwidth.

We simulate 5 types of periodic tasks T1-T5. Period of T3 and T4 is 1 minute, whereas T1, T2 and T5 are generated each 10 seconds. Table 6.8 shows the configurations of modelled generated tasks: TaskID, Sensor Type, period, complexity, Data size, Max response time, and its Classification. For example, T1 that performs "ECG classification" with a complexity of 1000MI, is classified as **Critical Analysis** and requires 16-36KB of measurement data, with 15s maximum response time [61]. Also, T2 that performs "speech analysis" is classified as **Data Analysis**, requires 500KB of measurement data, and has a complexity of 500MI with unlimited maximum response time [59]. The execution time

Table 6.8: Tasks Configuration

Ref	Task ID	Sensor(s) Type	Period (sec)	Complexity (MI)	Data size (KB)	Max response time(sec)	Classification
[61]	T1	ECG classification	10	500	36	15	critical analysis
[59]	T2	Audio	10	1000	500	unlimited	data analysis
[181]	T3	Temperature	60	10	10	500×10^{-3}	context management
[181]	T4	humidity	60	10	10	500×10^{-3}	context management
[182]	T5	ECG compression	10	211	240	500×10^{-3}	critical analysis

for a task is calculated as follows:

$$Execution\,Time(s) = \frac{Task\,complexity(MI)}{Processor\,Capacity(MIPS)} \tag{6.21}$$

Table 6.9 shows the results of makespan, network load, energy consumption (E_{total}), miss ratio, and number of handoffs , for allocating 133 generated tasks in full hospital occupancy (6 patients/room) condition. Missratio is measured by percentage of tasks that miss their maximum response time to the total number of tasks. Simulation time is set to 10 minutes, handoff threshold is set to -70dBm, allocation period α is set to 10s, and patient movement speed is set to 1.4m/s.

Table 6.9(a) demonstrates that makespan is equal to 10.4 seconds and does not exceed maximum required $MaxResponse$ for all the tasks. This is a good indicator on the ability of MobMBAR to be applied in critical application scenarios. It can also be noticed that the ratio of tasks missing their maximum response time is insignificant and equal to 0.2%. This articulates how MobMBAR can meet the required time constraints of the application. The network load which is the size of data transferred to the cloud for tasks execution, is equal to 637 MB. This load is a low volume traffic that can be easily provisioned with minimal latency and cost. Energy consumed by fog devices is equal to around 2458.2 joule where energy consumed by sink devices is equal to 0.8 Joule. We have to point out that adding a fog device to fog layer helps in enhancing the performance of the tasks processing, due to locally allocating the processing of tasks (on fog server) that were remotely allocated on the cloud.

Table 6.9: Case study results

(a) Fog layer with 4 fog devices

Measurement	**Value**
Makespan	10.4 seconds
Network load	637 MB
Energy consumption	2459 Joule
Miss ratio	0.002
Number of handoffs	89

(b) Fog layer with adding extra server

Measurement	**Value**
Makespan	7 seconds
Network load	412 MB
Energy consumption	1832.5 Joule
Miss ratio	0.001
Number of handoffs	87

Table 6.9(b) shows the performance results when adding Dell OptiPlex 780 fog server with 3GHz processor, 4GB RAM and 500GB storage. Results shows that MobMBAR provides makespan that achieves maximum response time of the tasks with reduction of energy consumption by about 25% and network load by 35%.

Chapter 7

Mobility-Enabled Privacy for Cloud-Fog IoT Healthcare

Privacy is defined as controlling the collection and usage of personal information during the lifetime of the application. In order to encourage patients to use the proposed architecture and to be relaxed when sharing his/her personal data, privacy must be provided throughout the different layers and modules. In this chapter, we tend to focus on the personal vital data collected and transmitted between the mobile sink devices and the gateways. To support this privacy, a strong authentication is required between both parties, the sink device and the gateway before and after the handoff operation, to prevent an attack risk if an attacker exploits this handoff as a security gap. Authentication proves and verifies the identities of the devices and hence helps in achieving privacy.

This chapter presents a mobility-enabled DTLS (Datagram Transport Layer Security)-based authentication scheme for securing the connection between the mobile sink device and the gateway when a handoff occurs. DTLS is considered as the standard security protocol for IoT constrained environments. It is an IP based end-to-end security protocol implemented over the transport layer. Based upon TLS (Transport Layer Security) protocol, DTLS share basic characteristics with TLS, while using UDP (User Datagram Protocol) instead of TCP (Transmission Control Protocol). DTLS is specified by IETF (Internet Engineering Task Force) and published in 2012 [1]. The first section summarizes the overall procedure of DTLS. The second section of this chapter reviews DTLS-based security in IoT, and the third section presents the proposed authentication procedure.

7.1 Introduction to DTLS Protocol

The execution of DTLS handshake is performed through interchanging six consecutive message flights between a client and a server. The client is the initiator of these messages. Each flight includes one or more communication messages. The first three flights are reserved for the detection and avoidance of Denial of Service (DoS) attack. The next 2 flights are used for the session generation through the authentication of peers and exchanging the security parameters. The final flight is used to sign the termination of the handshake and the launching of the session. The basic handshake is shown in figure 7.1 [1],[2]. Firstly, the client initiates the handshake by sending a $ClientHello$ message to the server

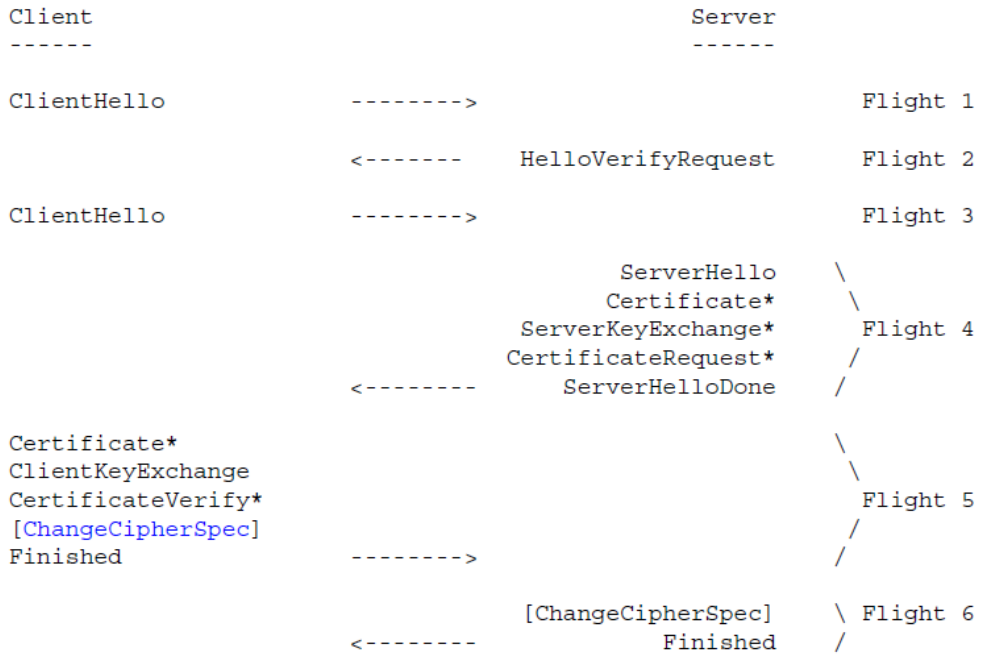


Figure 7.1: DTLS Overall Handshake [1],[2]

who responds with a $HelloVerifyRequest$ embedding a cookie to verify the client. Once the client replies with the same cookie in the second $ClientHello$ message, the server can exclude the client as being DOS attacker and continue the handshake, else the client is ignored. Following, the server start the exchange of security parameters with the client by sending $ServerHello$ message containing the cipher suites to be utilized, optional server digital certificate, and optional request for client certificate. The client then responds accordingly with the cipher suite chosen and its digital certificate if it was

requested, and an optional request for server certificate if it needs to have it while it was not sent by the server. The key exchange messages also include the exchange of pre-agreed keys needed to generate a final session key called master secret. The $ChangeCipherSpec$ message sent by both parties indicate the start of the session which means using the master secret generated for encrypting upcoming application messages.

One of three security modes can be used to accomplish this handshake and agree on session master secret; *pre-shared key (PSK)*, *raw public key (RPK)*, and *X.509 certificate based public key*. In *pre-shared key (PSK)*, a symmetric key based mutual authentication is performed based upon pre-distributed secret key. This mode is considered unscalable despite being lightweight. In *raw public key (RPK)*, asymmetric key authentication is performed. While certificate based public key mode is similar to *raw public key (RPK)*, an additional digital certificate is included. *Raw public key (RPK)* and *X.509 certificate based public key* both use Elliptic Curve Cryptography (ECC). Hence, they are more scalable and secure while they present higher computation overhead. The choice of the mode to use depends on the level of security needed in the application, in addition to the constraints of devices performing the handshake. *RPK* and *certificate based public key* are stronger than *PSK* despite that they require extra communication, computation, and storage overhead. *PSK* is more lightweight, fast, and is appropriate to most of IoT applications. However, it is non-scalable due to the need for pre-configuration of all communicating devices with the symmetric key.

However, as DTLS is a peer to peer authentication, when mobility exists, a new handshake between the new communicating devices is required. This in turn adds computational and communication overhead as well as extra delay which is unsuitable for critical applications such as healthcare. The next section proposes a scheme that maintains the secure authentication for IoT devices while transferring the connection between gateways, and eliminates the need for performing new full handshake.

7.2 Datagram Transport Layer Security (DTLS) in IoT

Several works have been proposed for enhancing the usage of DTLS in IoT. Authors in [183] work on a scheme for DTLS header compression to be more adaptable to energy constrained devices. Apart from working on the header structure, different articles worked on the platforms, the architecture of the

environment, or the handshake algorithm. In [184], [185], the authors propose an RSA keys certificate based DTLS handshake authentication. The work depends on the hardware RSA support existing in trusted platform module (TPM) for both computation and key storage. We found this work limited in the way that it needs a special chip on the IoT device due to the large key size in an RSA public key encryption. Consequently, the majority of other works employing DTLS based authentication in IoT relies on the elliptic curve cryptography. In [134], the authors delegate the handshake to a more powerful device which owns the resource constrained device. They use ECC keys certificate based for initial handshake and then utilize session resumption mechanism for reducing the overhead caused due to handshake re-establishments. In [186], the same authors update their architecture and replace the process of delegation from the gateways to be performed by one device called delegation server (DS) to overcome the mistrust in gateways. Despite being trusted and hence the approach is more secure, deploying one delegation server for performing the handshake drives towards scalability problem. The authors in [187] also use ECC key certificate based DTLS security. They utilize 6lowpan router (6LBR) to take part in securing the communication. It acts as an intermediary entity for authorizing both the client and the server. [188] performs a similar mechanism, but instead they employ a trust authority called trust anchor (TA) to authenticate and authorize the server to the client. They propose a method that is independent than the routing protocol used and also that reduces extra burden from the constrained device in addition to assuring the security. The authors work on two new versions of the preshared key and raw public key modes which are more scalable. In both, the client exchange the keys with TA and then it contacts the server. In addition, a new lightweight certificate called MAC certificate is proposed. The authors in [189] propose the usage of smart e-health gateways as an intermediate aiding entity in the security mechanism between the client and the server through applying a session resumption mechanism. They assume that when a mobile client is moved from the range of a smart e-health gateway to another, they don't have to perform an authentication as e-health gateways exchange security update messages about the clients. We found this method is non secure and threatens the security of both the client and the server. Adding to that, the authors doesn't reveal information about the security updates exchanged and hence, their contribution regarding mobility is not clear. A similar work is proposed in [190] where an IoT healthcare architecture also utilizes the

idea of gateways running DTLS authentication on behalf of the client representing the patient and also session resumption for reducing the overhead.

7.3 Proposed Mobility-Enabled Authentication

In this section, we propose a mobility-enabled DTLS-based authentication scheme for moving devices in IoT architectures. Our scheme is based on the session resumption functionality of TLS protocol in RFC 5077 [3]. However, in our work we consider an IoT device is able to resume its established secure session with a new gateway through sharing cryptographic security parameters between the gateways. In the following overview, we will subsequently present how the resumption is accomplished in the RFC.

7.3.1 Overview

In the specification of a session resumption without server side mechanism, a previously agreed upon session is allowed to be resumed between a peer of devices - a client and a server - that was initiated using a full handshake with the security mode selected. The following clarifies the full handshake with session resumption notification as shown in figure 7.2:

1. To notify the server of its preparation for a session resumption, the client extends the *client hello* message with an empty session ticket (flight 3).

2. If the server agrees to this request, it extends the *server hello* message with an empty session ticket (flight 4).

3. The handshake is continued using the security mode selected. In flight 6, the server generates and sends a complete session ticket to the client. The session ticket is a structure comprising multiple cryptographic security parameters concerning the server and the session state.

When the session is interrupted, terminated, or disconnected due to handoff, the client initiates an abbreviated handshake through sending the generated ticket to resume the session as shown in figure 7.3.

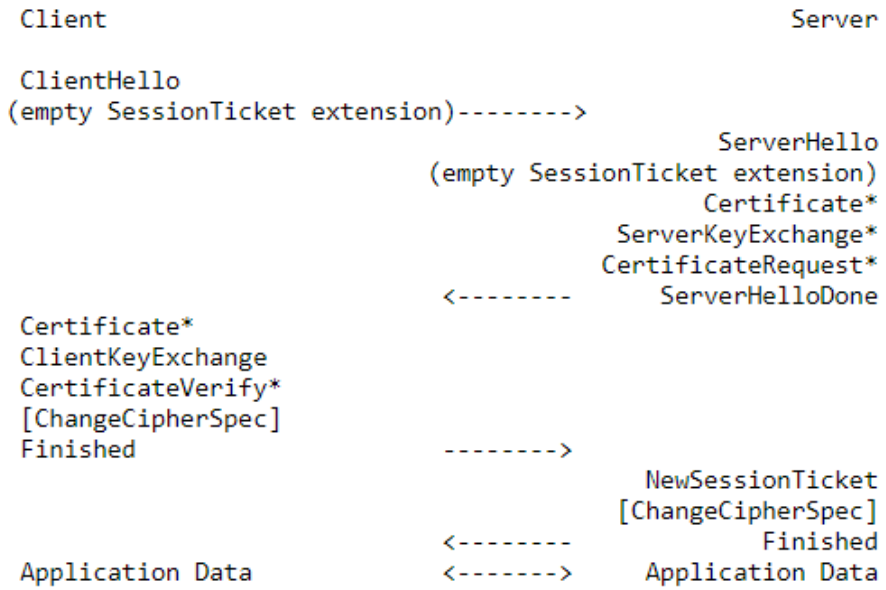


Figure 7.2: Full handshake with session resumption[3]

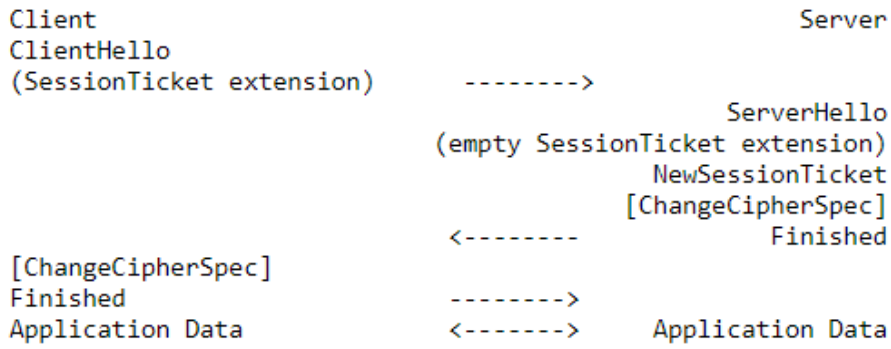


Figure 7.3: Abbreviated handshake [3]

In the following subsections, we propose a session resumption-based mobility-enabled DTLS handshake. Section B presents the IoT system architecture on which we apply our proposed mobility-enabled authentication, and section C presents detailed implementation of the modified version of handshake procedure.

7.3.2 Architecture

Figure 7.4 shows IoT system architecture on which we apply our proposed mobility-enabled authentication. In which, gateways layer comprising a number of interconnected gateways, and IoT devices layer comprising static and mobile IoT devices. The IoT devices utilize the gateways for processing and forwarding the collected data. The gateway is connected to single/multiple IoT devices, but the IoT device is connected to one gateway at a time. For our proposed mobility-enabled authentication scheme, we assume the following:

- After network setup, each pair of gateways (gi, gj) mutually authenticate each other using certificate based DTLS and agree upon a key key_gi_gj for encrypting their exchanged messages.
- Single hop communication between IoT device (client) and gateway (server).
- Upon entering the transmission range of a gateway, mobile IoT device initiates a DTLS based handshake with the gateway so that they mutually authenticate each other.
- Raw public key (RPK) DTLS mode is implemented when performing the handshake between IoT device and gateway. It is lighter than certificate based DTLS mode and more secure than preshared key (PSK) mode, and it suits the capabilities of constrained IoT devices.

As it can be seen from figure 7.4, considering a scenario that an IoT device such as a medical sensor hanged by patient has mutually authenticated with home gateway. When the patient moves, this medical sensor can change its connection from home gateway to visited gateway. Handoff was previously explained in chapter 6. In order to overcome the lack of security caused as a result of the handoff and carry on with the authentication between IoT device and visisted gateway, we propose mobility-enabled DTLS-based authentication scheme described in the next section.

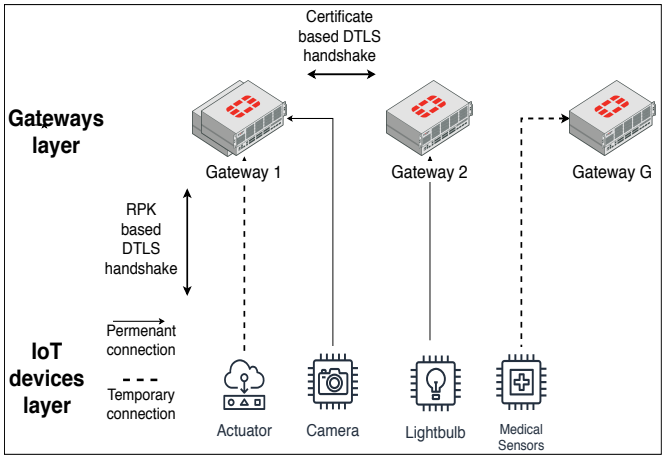


Figure 7.4: Proposed scheme architecture

7.3.3 DTLS-based Handshake

The implementation of the proposed scheme is based on communication messages that have the following structures shown in figure 7.5 created by the gateway.

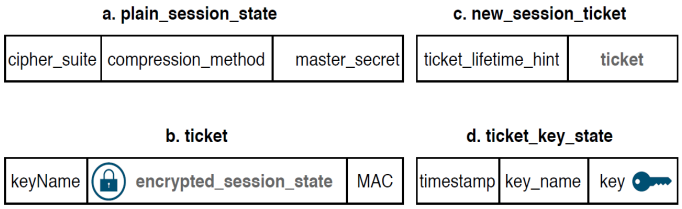


Figure 7.5: Communication messages of proposed scheme

- $plain\_session\_state$: It holds the state of the session established between an IoT device and a gateway. It consists of : 1) $cipher\_suite$: agreed upon between an IoT device and gateway, 2) $compression\_method$: the technique used to compress the data, and 3) $master\_secret$: the key that is used to encrypt the data during the session established between IoT device and gateway.

- $ticket$: It encapsulates the session state to be forwarded to the IoT device. Ticket structure is composed of three fields: 1) $encrypted\_session\_state$ which is a session state encrypted using a randomly generated key by the gateway at the startup of the device, 2) $keyName$: the name

of the key used to encrypt the session state, and 3) MAC: message authentication code to authenticate the encrypted session state.

- $new\_session\_ticket$: The new session ticket is the generated ticket sent to the IoT device associated with a hint on the period during which the ticket is valid ($ticket\_lifetime\_hint$).

- $ticket\_key\_state$: It holds information concerning the randomly key generated by the gateway at the startup of device. It consists of the name of the key (key_name) used to encrypt the session state between the IoT device and the gateway, the key itself, and a timestamp of this key state.

The procedure of proposed mobility-enabled authentication scheme is presented in figure 7.6 where:

1. A full RPK based DTLS handshake with session resumption is performed following the sequence described in subsection 7.3.1. Where an IoT device sends *client hello* message to the gateway with an empty session ticket extension, and accordingly, the gateway replies with a *server hello* message with an empty session ticket extension. Hence, the cipher suite is set to *TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8* (flights 1-5).

   The first segment of the cipher suite is the key agreement protocol. Elliptic curve Diffie-Hellman Ephemeral (ECDHE) and Elliptic Curve Digital Signature Algorithm (ECDSA) key agreement algorithms allow IoT device and gateway, each having an elliptic curve public-private key pair, to establish a shared secret. Elliptic curve cryptography (ECC) is able to provide high cryptographic strength with much smaller key sizes which distinguishes it than other public key cryptography algorithms e.g. RSA. The second segment of the cipher suite defines the encryption algorithm utilized during the session. Advanced Encryption Standard (AES) with 128-bit key is very popular cryptographic scheme in IoT due to its robustness in addition to it simple implementation which is suitable for the constrained devices. Finally, CCM_8 (Counter with CBC-MAC with an 8-Octet) segment identifies the usage of AES-CCM mode which generates a MAC to achieve integrity.

2. The gateway encrypts the session state $plain\_session\_state$ with a randomly generated key key

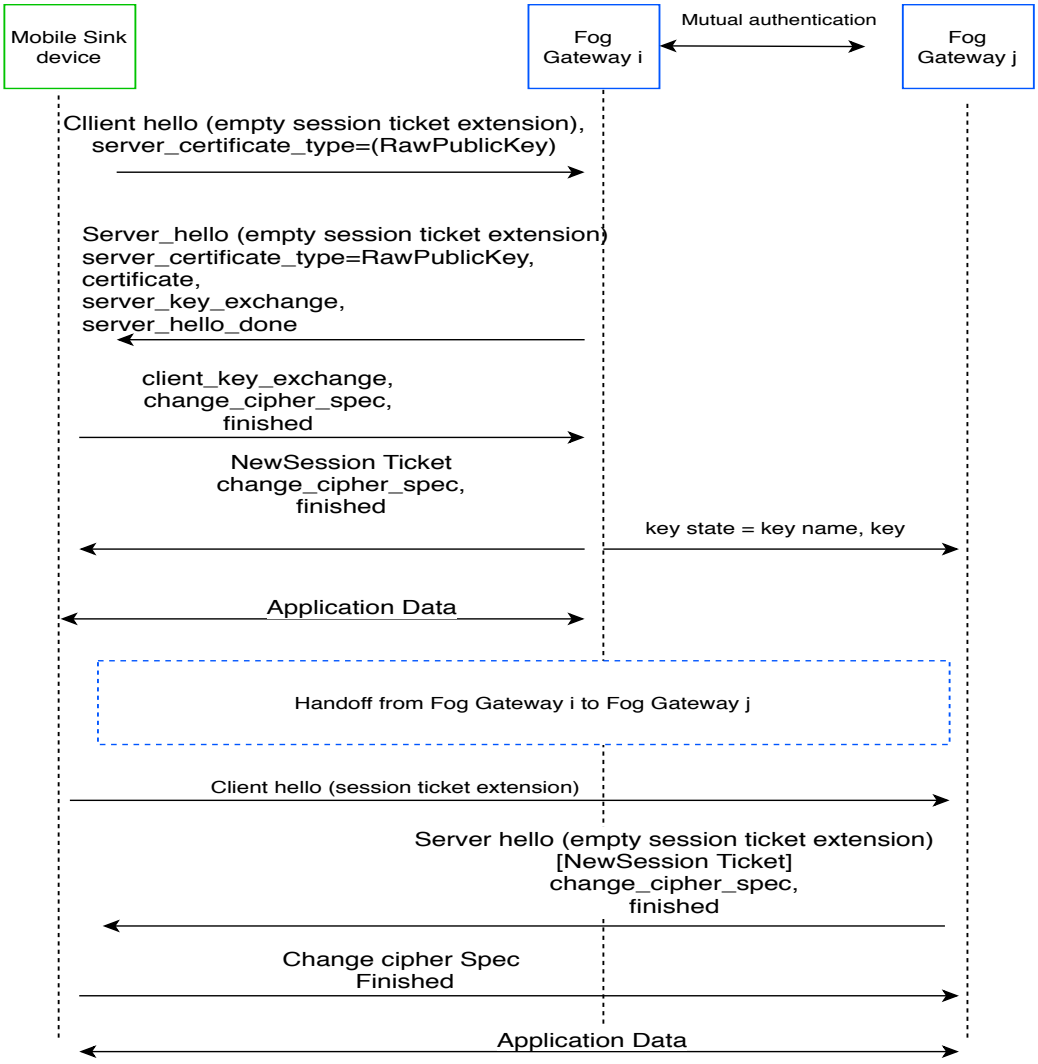


Figure 7.6: Proposed DTLS-based Handshake

at the startup of the device, and sends the complete session ticket $ticket$ along with the name of the key key_name to the IoT device (flight 6).

3. To allow the mobile IoT device to be able to resume the connection with other gateway, the gateway sends the complete state of the key $ticket\_key\_state$ to the mutually authenticated gateways encrypted using key_gi_gj.

Upon the handoff of the mobile IoT device between gateway i and gateway j:

1. The mobile IoT device starts DTLS handshake by sending the *client_hello* message to the visited gateway along with the session ticket $new\_session\_ticket$.

2. The visited gateway firstly checks the lifetime validity of $new\_session\_ticket$. Next, it extracts the ticket $ticket$ and checks the existence of $keyName$ in its list of $ticket\_key\_state$ shared by the neighboring gateways. If it finds a match, it replies to the IoT device with a verification message (an empty session ticket extension to the *server_hello*).

3. Then, the visited gateway decrypts the $encrypted\_session\_state$ of the ticket $ticket$ using the inferred key key in $ticket\_key\_state$.

4. Both the visited gateway and IoT device change their cipher specifications according to the cryptographic security parameters of the decrypted session state $plain\_session\_state$: $cipher\_suite$, $compression\_method$, $master\_secret$.

The proposed scheme maintains the secure authentication while transferring the connection among gateways, and eliminates the need for performing new full handshake with its key exchange messages.

7.4 Performance Evaluation

In this section, we evaluate the performance of proposed mobility-enabled DTLS-based authentication scheme. The proposed scheme is implemented utilizing Contiki Cooja simulation tool which is a network simulator designed for constrained devices. The simulation is running on VM setup on 3.2 GHz laptop. We simulate 16 MHz WiSMote devices [191] equipped with CC2520 IEEE 802.15.4 RF

Table 7.1: WiSMote specifications

Parameter	Value
Frequency	18 MHz
Flash	256 KB
SRAM	16 KB
ADC	12-bit

transceiver [192]. Table 7.1 shows the specifications of a WiSMote device. We make use of tinyDTLS which is a Datagram Transport Layer Security (DTLS) library supporting the CoAP oriented cipher suite [193]. We build certificate-based DTLS, and a session resumption without server side state algorithm on TinyDTLS code. We utilize Aaron Gifford's Implementations of NIST public key operations and hash algorithms [194].

7.4.1 Evaluation metrics

To evaluate the efficiency of the proposed scheme, we measure and compare the performance of mutual authentication process between IoT device and visited gateway when handoff occurs using: the proposed scheme based on session resumption handshake against new session establishment handshake. We measure the following four metrics:

- Handshake latency: defines the latency of authentication in case of a new session established by full handshake protocol, or session resumed by proposed scheme, when handling handoff .
- Processing time: refers to in-node computation overhead for IoT device and gateway in case of implementing the full handshake, and implementing the proposed session resumption-based handshake. To measure the processing time, a timer is started on the transmission of client hello message until the reception of finished message.
- Energy consumption: indicates the total communication and computation energy consumed by IoT device and the gateway, when implementing both cases: full handshake and proposed ses-

sion resumption-based handshake. $Powertrace$ program in Contiki Cooja is used to evaluate the total energy consumption.

- Memory overhead: is the usage of read-only memory (ROM) and random-access memory (RAM) for the IoT device and the gateway in the cases of full handshake and proposed session resumption-based handshake. To evaluate the usage of memory and size of code, $GNUsize$ is used to measure memory sizes.

7.4.2 Results

Table 7.2 shows the andhshake latency created by full handshake protocol and proposed scheme in the case of a handoff between gateway i and visited gateway j. The proposed scheme speeds-up the mutual authentication process between IoT device and visited gateway j by 93% compared to full handshake, due to the reduction in the number of flight messages exchanged.

Table 7.2: Handshake latency

Measurement	**Full handshake**	**Proposed scheme**	**Improvement**
Handshake latency	600 s	40 ms	93%

Tables 7.3, 7.4 show the performance results of handling handoff using full handshake and the proposed session resumption-based handshake for both IoT device and gateway respectively. Table 7.3 shows that the processing time for the proposed session resumption-based handshake at IoT device is reduced by about 99.7%, due to the elimination of cryptographic key and hash operations required to perform the authentication and generate the master secret.

Consequently, it can be noticed from table 7.3 that the energy consumption for the proposed session resumption-based handshake at IoT device is reduced by the same ratio to reach 39 mJ for computations. Regarding communication energy consumption for proposed scheme, it is observed that the transmit energy consumed at the IoT device is 1.8 mJ while the receive energy is 2.4 mJ, with energy saving 4.5%, and 7% respectively compared to full handshake, due to minimal number of sent and received messages for resuming the session. Whereas full handshake requires additional messages for

Table 7.3: Performance Results for IoT device

Measurement	**Full handshake**	**Proposed scheme**	**Improvement**
Processing time	192 s	703 ms	99.7%
Energy consumption	processing (10 J)	processing (39 mJ)	99.6%
	radio tx (40 mJ)	radio tx (1.8 mJ)	4.5%
	radio rx (33 J)	radio rx (2.4 J)	7%
RAM overhead	0.43 kB	0.44 kB	-
ROM overhead	86.6 kB	87 kB	-

Table 7.4: Performance Results for Gateway

Measurement	**Full handshake**	**Proposed scheme**	Improvement
Processing time	196 s	1156 ms	99.8%
Energy consumption	processing (10.7 J)	processing (63 mJ)	99.4%
	radio tx (23 mJ)	radio tx (1.2 mJ)	5%
	radio rx (32 J)	radio rx (2.1 J)	6.5%
RAM overhead	0.46 kB	0.47 kB	-
ROM overhead	88 kB	88 kB	-

key exchanging between IoT device and visited gateway. Table 7.3 also shows ROM and RAM usage at IoT device for the full handshake against the session resumption-based handshake. The session resumption-based handshake exhibits larger memory footprint of about 400 bytes as it includes additional code for handling the session tickets. no significant improvement is observed in RAM overhead.

Table 7.4 shows a reduction by about 99.8% in the processing time at gateway for the proposed session resumption-based handshake compared to the full handshake. Also, computation overhead at the gateway is reduced by about 99.4% for the proposed session resumption-based handshake compared to the full handshake. In addition, the proposed session resumption-based handshake achieves 5% and 6.5% reduction in communication energy consumption in transmit and receive mode respectively compared to full handshake. Close measurement values for memory usage are noticed at the

gateway for both full handshake and the proposed session resumption-based handshake.

7.4.3 Security analysis

The proposed scheme provides confidentiality, integrity, forward secrecy through applying AES-CCM cryptography encryption, HMAC-SHA-256 on the message encrypted, and applying Elliptic Curve Cryptography (ECC) for generating the session key respectively. AES (Advanced Encryption Standard) is known for being strong symmetric encryption technique applicable to constrained IoT devices. SHA-256 (Secure Hash Algorithm 256-bit) cryptographic hash algorithm produces irreversible and unique hash to guarantee no altering of the data while being simple and appropriate to IoT devices at the same time.

We analyze the proposed scheme for most common IoT attacks: Denial of Service (DOS), Eavesdropping, and impersonation attacks. An attacker may launch a denial of service attack through flooding the IoT gateway with client hello messages so it becomes unresponsive. This is forbidden utilizing the hello verify request message sent by the IoT gateway which then waits a valid cookie from the IoT device. If this valid cookie is not received, the DTLS state initialization is never started. If an attacker performs an eavesdropping on a session ticket sent by an IoT device to visited IoT gateway, it won't be able to obtain the master secret as it is encrypted using a random key generated by the home IoT gateway. Even this key is exchanged in an encrypted form between the IoT gateways. In the same way, if an attacker impersonates a visited IoT gateway receiving a session ticket from an IoT device to receive the data directly continuously without eavesdropping, it won't be able to decrypt the ticket as it is not connected, neither performed handshake with its neighbor IoT gateways.

The proposed scheme is able to authenticate between a mobile IoT device and visited gateway while reducing the overhead of authentication. Compared to other works, e.g. [189] where no authentication is performed when the mobile device changes its gateway. This lack of authentication is inadequate with the sensitivity nature and large amount of the data gathered and processed by the gateways and the fog devices.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

IoT for healthcare is a promising approach for long-term ubiquitous daily activity and health monitoring. It endorses the unobtrusive biomedical devices to gather patients' vital signs data and activities in-home and outside it.

This research proposed a dynamic Cloud-Fog inter-operable IoT architecture for healthcare. The proposed architecture supports the mobility of the patients as well as the diversity of the medical cases. The features of the individual modules are discussed and the interconnection between the different underlying modules and tiers is explained. Firstly, Modified BAlance Reduce (MBAR) task scheduling and allocation algorithm is proposed to effectively balance healthcare tasks distribution. Next, (MBAR) is extended and adapted to support the mobility of patients, resulting in implementing a mobility-aware Modified BAlance Reduce (MobMBAR) for IoT healthcare architecture in cloud-fog computing paradigm. MobMBAR performs dynamic balanced healthcare tasks distribution between the cloud and fog devices. It is a data locality-based approach that depends on changing the location where the data is computed to where it actually resides. It also considers the priorities of tasks represented in the classification (Class) and maximum response time MaxResponse. As such, as long as the task has the highest classification and lowest MaxResponse, it has higher scheduling priority. The proposed approach is able to guarantee the QoS through reliable on-time execution of healthcare tasks and allows the execution of heterogeneous healthcare tasks featured by different processing rates, data

sizes, and number.

We conduct a comparative study against different approaches to evaluate the proposed work using different simulation experiments. The experimental results illustrate that MobMBAR have distinct lower makespan compared to HealthEdge, HEFT, and Cloud-Only. In terms of network load performance, MobMBAR shows network traffic load that is lower than HEFT and Cloud-Only. These results prove that MobMBAR outperforms HealthEdge, HEFT, Cloud-Only in terms of latency preserving, while maintaining acceptable network traffic load. In addition, a case study that uses the layout of a hospital building in Chicago is presented to evaluate MobMBAR in real conditions. Results show that makespan does not exceed maximum required MaxResponse for all generated tasks, ratio of tasks missing their maximum response time is insignificant, and low volume traffic that can be easily provisioned with minimal latency and cost.

This research also proposed a mobility-enabled DTLS-based authentication method for moving IoT devices in indoor IoT architectures. The proposed method makes use of the session resumption feature of the DTLS standard. Utilizing the proposed methodology, additional handshake for the same IoT device in the same network is eliminated and hence extra time, power consumption, and energy are saved. The proposed method is evaluated in terms of handshake time, processing time, energy consumption, and memory overhead. The results prove the efficiency of the proposed method and its good impact when implemented in mobile IoT architectures.

8.2 Future Work

The seamless integration of cloud computing and fog computing for healthcare systems introduces several advantages, which include richer functionalities, services and performance efficiency and facilitating the exchange of electronic medical records among hospitals and clinics. To support scalability, high mobility and fault tolerance of the proposed scheme, we plan to extend the proposed architecture by replicating multiple fog broker nodes. We intend to propose a distributed scheduling version to enhance the scalability and to minimize execution and response time. Also we plan to implement a real experimental prototype so that the proposed model can be evaluated and real results can be verified to actual healthcare cases.

The proposed architecture is open to multiple research investigations that can promote its efficiency. This can include dual or multiple optimization metrics for the scheduling and allocation. For instance, minimizing the cost of utilizing the cloud resources can be presented as an optimization metric in addition to the latency. Obviously, this will depend on the health conditions and emergency levels of the patients supported and the characteristics of the tasks beside the monetary budget of the health institution. Open research directions can include energy efficient algorithms that tend to minimize energy consumption of the IoT constrained devices. They can be implemented at the IoT devices and sink layers where constrained IoT devices exist. For example, energy harvesting for the IoT devices and sink layers through data collection can be discussed and examined. Sensor fusion researches can be explored and utilized in these two layers. Sensor fusion can be used as a pre-processing phase before the data is sent to the fog and cloud layers. Adaptive fusion technique can be implemented for saving energy of sensors. In addition, adaptive data sensing which means to trigger sensors periodically instead of continously based on the medical case and the status of the patient can be investigated. Open security directions involve proposing a scheme for storage of encrypted data to prevent data breach in case of malicious fog node. In addition, cloud security through advanced tools such as cloud access security broker (CASB) systems is of an important interest to protect health private data while safely using the cloud.

Bibliography

[1] Yakov Rekhter and N. Modadugu. Datagram Transport Layer Security Version 1.2. RFC 6347, RFC Editor, January 2012.

[2] Hannes Tschofenig and Thomas Fossati. Transport layer security (tls)/datagram transport layer security (dtls) profiles for the internet of things. In *RFC 7925*. Internet Engineering Task Force, 2016.

[3] P. Eronen J. Salowey, H. Zhou and H. Tschofenig. Transport Layer Security (TLS) Session Resumption without Server-Side State. RFC 5077, RFC Editor, January 2008.

[4] New IDC Forecast:The Growth in Connected IoT Devices. `http://idc.com`. Accessed: 2020-07-24.

[5] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

[6] Przemyslaw Woznowski, Alison Burrows, Tom Diethe, Xenofon Fafoutis, Jake Hall, Sion Hannuna, Massimo Camplani, Niall Twomey, Michal Kozlowski, Bo Tan, et al. SPHERE: A sensor platform for healthcare in a residential environment. In *Designing, Developing, and Facilitating Smart Cities*, pages 315–333. Springer, 2017.

[7] Robert SH Istepanian, Sijung Hu, Nada Y Philip, and Ala Sungoor. The potential of internet of m-health things "m-iot" for non-invasive glucose level sensing. In *2011 Annual International*

Conference of the IEEE Engineering in Medicine and Biology Society, pages 5264–5266. IEEE, 2011.

[8] Vandana Milind Rohokale, Neeli Rashmi Prasad, and Ramjee Prasad. A cooperative internet of things(IoT) for rural healthcare monitoring and control. In *2011 2nd international conference on wireless communication, vehicular technology, information theory and aerospace & electronic systems technology (Wireless VITAE)*, pages 1–6. IEEE, 2011.

[9] Antonio J Jara, Miguel A Zamora-Izquierdo, and Antonio F Skarmeta. Interconnection framework for mHealth and remote monitoring based on the internet of things. *IEEE Journal on Selected Areas in Communications*, 31(9):47–65, 2013.

[10] Mohd Fadlee A Rasid, Wan Mardiana Wan Musa, Nurul Ashikin Abdul Kadir, Anas M Noor, Farid Touati, Waiser Mehmood, Lazhar Khriji, Asma Al-Busaidi, and A Ben Mnaouer. Embedded gateway services for Internet of Things applications in ubiquitous healthcare. In *2014 2nd International Conference on Information and Communication Technology (ICoICT)*, pages 145–148. IEEE, 2014.

[11] Rajiv Chakravorty et al. MobiCare: A programmable service architecture for mobile medical care. *Proceedings of UbiCare*, pages 532–536, 2006.

[12] MPR Sai Kiran, Pachamuthu Rajalakshmi, and Amit Acharyya. Context predictor based sparse sensing technique and smart transmission architecture for IoT enabled remote health monitoring applications. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4151–4154. IEEE, 2014.

[13] Geng Yang, Li Xie, Matti Mäntysalo, Xiaolin Zhou, Zhibo Pang, Li Da Xu, Sharon Kao-Walter, Qiang Chen, and Li-Rong Zheng. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. *IEEE transactions on industrial informatics*, 10(4):2180–2191, 2014.

[14] Prabal Verma and Sandeep K Sood. Fog assisted-iot enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal*, 5(3):1789–1796, 2018.

[15] Shirin Enshaeifar, Payam Barnaghi, Severin Skillman, Andreas Markides, Tarek Elsaleh, Sahr Thomas Acton, Ramin Nilforooshan, and Helen Rostill. The internet of things for dementia care. *IEEE Internet Computing*, 22(1):8–17, 2018.

[16] Amir-Mohammad Rahmani, Nanda Kumar Thanigaivelan, Tuan Nguyen Gia, Jose Granados, Behailu Negash, Pasi Liljeberg, and Hannu Tenhunen. Smart e-health gateway: Bringing intelligence to internet-of-things based ubiquitous healthcare systems. In *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 826–834, 2015.

[17] Priyan Malarvizhi Kumar, S Lokesh, R Varatharajan, Gokulnath Chandra Babu, and P Parthasarathy. Cloud and iot based disease prediction and diagnosis system for healthcare using fuzzy neural classifier. *Future Generation Computer Systems*, 86:527–534, 2018.

[18] Prabal Verma and Sandeep K Sood. Cloud-centric iot based disease diagnosis healthcare framework. *Journal of Parallel and Distributed Computing*, 116:27–38, 2018.

[19] Shreshth Tuli, Nipam Basumatary, Sukhpal Singh Gill, Mohsen Kahani, Rajesh Chand Arya, Gurpreet Singh Wander, and Rajkumar Buyya. Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments. *Future Generation Computer Systems*, 104:187–200, 2020.

[20] Agnes Grünerbl, Amir Muaremi, Venet Osmani, Gernot Bahle, Stefan Oehler, Gerhard Tröster, Oscar Mayora, Christian Haring, and Paul Lukowicz. Smartphone-based recognition of states and state changes in bipolar disorder patients. *IEEE Journal of Biomedical and Health Informatics*, 19(1):140–148, 2014.

[21] Tuan Nguyen Gia, Victor Kathan Sarker, Igor Tcarenko, Amir M Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Energy efficient wearable sensor node for iot-based fall detection systems. *Microprocessors and Microsystems*, 56:34–46, 2018.

[22] Yueng Santiago Delahoz and Miguel Angel Labrador. Survey on fall detection and fall prevention using wearable and external sensors. *Sensors*, 14(10):19806–19842, 2014.

[23] Ramesh Rajagopalan, Irene Litvan, and Tzyy-Ping Jung. Fall prediction and prevention systems: recent trends, challenges, and future research directions. *Sensors*, 17(11):2509, 2017.

[24] Anita Ramachandran and Anupama Karuppiah. A survey on recent advances in wearable fall detection systems. *BioMed research international*, 2020.

[25] Lofti A Zadeh. Information and control. *Fuzzy sets*, 8(3):338–353, 1965.

[26] Tom Diethe, Niall Twomey, Meelis Kull, Peter Flach, and Ian Craddock. Probabilistic sensor fusion for ambient assisted living. *arXiv preprint arXiv:1702.01209*, 2017.

[27] Nidal AlBeiruti and Khalid Al Begain. Using hidden markov models to build behavioural models to detect the onset of dementia. In *Computational Intelligence, Communication Systems and Networks (CICSyN), 2014 Sixth International Conference on*, pages 18–26. IEEE, 2014.

[28] Paulo Armando Cavalcante Aguilar, Jerome Boudy, Dan Istrate, Bernadette Dorizzi, and Joao Cesar Moura Mota. A dynamic evidential network for fall detection. *IEEE journal of biomedical and health informatics*, 18(4):1103–1113, 2014.

[29] Agnes Grünerbl, Amir Muaremi, Venet Osmani, Gernot Bahle, Stefan Oehler, Gerhard Tröster, Oscar Mayora, Christian Haring, and Paul Lukowicz. Smartphone-based recognition of states and state changes in bipolar disorder patients. *IEEE Journal of Biomedical and Health Informatics*, 19(1):140–148, 2015.

[30] Hamid Medjahed, Dan Istrate, Jerome Boudy, Jean-Louis Baldinger, and Bernadette Dorizzi. A pervasive multi-sensor data fusion for smart home healthcare monitoring. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 1466–1473. IEEE, 2011.

[31] Raffaele Gravina, Parastoo Alinia, Hassan Ghasemzadeh, and Giancarlo Fortino. Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges. *Information Fusion*, 35:68–80, 2017.

[32] Furqan Alam, Rashid Mehmood, Iyad Katib, Nasser Albogami, and Aiiad Albeshri. Data Fusion and IoT for Smart Ubiquitous Environments: A Survey. *IEEE Access*, 2017.

[33] Shilpa Gite and Himanshu Agrawal. On context awareness for multisensor data fusion in iot. In *Proceedings of the Second International Conference on Computer and Communication Technologies*, pages 85–93. Springer, 2016.

[34] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, and Francisco Flórez-Revuelta. From data acquisition to data fusion: a comprehensive review and a roadmap for the identification of activities of daily living using mobile devices. *Sensors*, 16(2):184, 2016.

[35] Meisong Wang, Charith Perera, Prem Prakash Jayaraman, Miranda Zhang, Peter Strazdins, RK Shyamsundar, and Rajiv Ranjan. City data fusion: Sensor data fusion in the internet of things. In *The Internet of Things: Breakthroughs in Research and Practice*, pages 398–422. IGI Global, 2017.

[36] Shahina Begum, Shaibal Barua, and Mobyen Uddin Ahmed. Physiological sensor signals classification for healthcare using sensor data fusion and case-based reasoning. *Sensors*, 14(7):11770–11785, 2014.

[37] Khaled M Gharaibeh and Orobah Al-Momani. Fuzzy logic-based decision-making system for asthma attack detection. *International Journal of Biomedical Engineering and Technology*, 15(2):155–172, 2014.

[38] Carol Habib, Abdallah Makhoul, Rony Darazi, and Raphaël Couturier. Multisensor data fusion and decision support in wireless body sensor networks. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 708–712. IEEE, 2016.

[39] Won-Jae Yi, Oishee Sarkar, Sivisa Mathavan, and Jafar Saniie. Wearable sensor data fusion for remote health assessment and fall detection. In *Electro/Information Technology (EIT), 2014 IEEE International Conference on*, pages 303–307. IEEE, 2014.

[40] Won-Jae Yi, Oishee Sarkar, Sivisa Mathavan, and Jafar Saniie. Design flow of wearable heart monitoring and fall detection system using wireless intelligent personal communication node. In *Electro/Information Technology (EIT), 2015 IEEE International Conference on*, pages 314–319. IEEE, 2015.

[41] Gregory Koshmak, Maria Linden, and Amy Loutfi. Dynamic bayesian networks for context-aware fall risk assessment. *Sensors*, 14(5):9330–9348, 2014.

[42] Abdur Rahim Mohammad Forkan, Ibrahim Khalil, Zahir Tari, Sebti Foufou, and Abdelaziz Bouras. A context-aware approach for long-term behavioural change detection and abnormality prediction in ambient assisted living. *Pattern Recognition*, 48(3):628–641, 2015.

[43] Jean-Louis Baldinger, Jérôme Boudy, Bernadette Dorizzi, Jean-Pierre Levrey, Rodrigo Andreao, Christian Perpère, François Delavault, François Rocaries, Christophe Dietrich, and Alain Lacombe. Tele-surveillance System for Patient at Home: the MEDIVILLE system. In *International Conference on Computers for Handicapped Persons*, pages 400–407. Springer, 2004.

[44] Hamid Medjahed. *Distress situation identification by multimodal data fusion for home healthcare telemonitoring*. PhD thesis, Institut National des Télécommunications, 2010.

[45] Roberto Paoli, Francisco J Fernández-Luque, Ginés Doménech, Félix Martínez, Juan Zapata, and Ramón Ruiz. A system for ubiquitous fall monitoring at home via a wireless sensor network and a wearable mote. *Expert systems with applications*, 39(5):5566–5575, 2012.

[46] Yoann Charlon, Nicolas Fourty, Walid Bourennane, and Eric Campo. Design and evaluation of a device worn for fall detection and localization: Application for the continuous monitoring of risks incurred by dependents in an alzheimer's care unit. *Expert Systems with Applications*, 40(18):7316–7330, 2013.

[47] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[48] Mohammad Shojafar, Zahra Pooranian, Paola G Vinueza Naranjo, and Enzo Baccarelli. Flaps: bandwidth and delay-efficient distributed data searching in fog-supported p2p content delivery networks. *The journal of supercomputing*, 73(12):5239–5260, 2017.

[49] Yang Cai, Angelo Genovese, Vincenzo Piuri, Fabio Scotti, and Mel Siegel. Iot-based architectures for sensing and local data processing in ambient intelligence: Research and industrial trends. In *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6. IEEE, 2019.

[50] Xiang Sun and Nirwan Ansari. Edgeiot: Mobile edge computing for the internet of things. *IEEE Communications Magazine*, 54(12):22–29, 2016.

[51] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95, 2017.

[52] Jorge Gomez, Byron Oviedo, and Emilio Zhuma. Patient monitoring system based on internet of things. *Procedia Computer Science*, 83:90–97, 2016.

[53] Mingzhe Jiang, Tuan Nguyen Gia, Arman Anzanpour, Amir-Mohammad Rahmani, Tomi Westerlund, Sanna Salanterä, Pasi Liljeberg, and Hannu Tenhunen. Iot-based remote facial expression monitoring system with semg signal. In *Sensors Applications Symposium (SAS), 2016 IEEE*, pages 1–6. IEEE, 2016.

[54] Yang-Yen Ou, Po-Yi Shih, Yu-Hao Chin, Ta-Wen Kuan, Jhing-Fa Wang, and Shao-Hsien Shih. Framework of ubiquitous healthcare system based on cloud computing for elderly living. In *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*, pages 1–4. IEEE, 2013.

[55] Ee-May Fong and Wan-Young Chung. Mobile cloud-computing-based healthcare service by noncontact ecg monitoring. *Sensors*, 13(12):16451–16473, 2013.

[56] Zulqarnain Rashid, Umar Farooq, Jae-Keun Jang, and Seung-Hun Park. Cloud computing aware ubiquitous health care system. In *E-Health and Bioengineering Conference (EHB), 2011*, pages 1–4. IEEE, 2011.

[57] Razvan Craciunescu, Albena Mihovska, Mihail Mihaylov, Sofoklis Kyriazakos, Ramjee Prasad, and Simona Halunga. Implementation of fog computing for reliable e-health applications. In *Signals, Systems and Computers, 2015 49th Asilomar Conference on*, pages 459–463. IEEE, 2015.

[58] Yu Cao, Songqing Chen, Peng Hou, and Donald Brown. Fast: A fog computing assisted distributed analytics system to monitor fall for stroke mitigation. In *Networking, Architecture and Storage (NAS), 2015 IEEE International Conference on*, pages 2–11. IEEE, 2015.

[59] Admir Monteiro, Harishchandra Dubey, Leslie Mahler, Qing Yang, and Kunal Mankodiya. Fit a fog computing device for speech teletreatments. *arXiv preprint arXiv:1605.06236*, 2016.

[60] Xavier Masip-Bruin, Eva Marín-Tordera, Albert Alonso, and Jordi Garcia. Fog-to-cloud computing (F2C): the key technology enabler for dependable e-health services deployment. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2016 Mediterranean*, pages 1–5. IEEE, 2016.

[61] Harishchandra Dubey, Jing Yang, Nick Constant, Amir Mohammad Amiri, Qing Yang, and Kunal Makodiya. Fog data: Enhancing telehealth big data through fog computing. In *Proceedings of the ASE BigData & SocialInformatics 2015*, page 14. ACM, 2015.

[62] Flavio Bonomi, Rodolfo Milito, Preethi Natarajan, and Jiang Zhu. Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments*, pages 169–186. Springer, 2014.

[63] Amir Vahid Dastjerdi and Rajkumar Buyya. Fog computing: Helping the internet of things realize its potential. *Computer*, 49(8):112–116, 2016.

[64] Mohammad Aazam, Sherali Zeadally, and Khaled A Harras. Offloading in fog computing for iot: review, enabling technologies, and research opportunities. *Future Generation Computer Systems*, 87:278–289, 2018.

[65] Pinal Salot. A survey of various scheduling algorithm in cloud computing environment. *International Journal of Research in Engineering and Technology*, 2(2):131–135, 2013.

[66] Abdul Hameed, Alireza Khoshkbarforoushha, Rajiv Ranjan, Prem Prakash Jayaraman, Joanna Kolodziej, Pavan Balaji, Sherali Zeadally, Qutaibah Marwan Malluhi, Nikos Tziritas, Abhinav Vishnu, et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98(7):751–774, 2016.

[67] Simsy Xavier and SJ Lovesum. A survey of various workflow scheduling algorithms in cloud environment. *International Journal of Scientific and Research Publications*, 3(2), 2013.

[68] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, and Dan Wang. Cloud task scheduling based on load balancing ant colony optimization. In *2011 sixth annual ChinaGrid conference*, pages 3–9. IEEE, 2011.

[69] Hong-Min Chu, Shao-Wen Yang, Padmanabhan Pillai, and Yen-Kuang Chen. Scheduling in visual fog computing: NP-completeness and practical efficient solutions. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[70] Nitinder Mohan and Jussi Kangasharju. Edge-fog cloud: A distributed cloud for internet of things computations. In *2016 Cloudification of the Internet of Things (CIoT)*, pages 1–6. IEEE, 2016.

[71] Xuan-Qui Pham and Eui-Nam Huh. Towards task scheduling in a cloud-fog computing system. In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4. IEEE, 2016.

[72] Xuan-Qui Pham, Nguyen Doan Man, Nguyen Dao Tan Tri, Ngo Quang Thai, and Eui-Nam Huh. A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. *International Journal of Distributed Sensor Networks*, 13(11):1550147717742073, 2017.

[73] Suraj Pandey, Linlin Wu, Siddeswara Mayura Guru, and Rajkumar Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *2010 24th IEEE international conference on advanced information networking and applications*, pages 400–407. IEEE, 2010.

[74] Mingyue Feng, Xiao Wang, Yongjin Zhang, and Jianshi Li. Multi-objective particle swarm optimization for resource allocation in cloud computing. In *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, volume 3, pages 1161–1165. IEEE, 2012.

[75] Lizheng Guo, Shuguang Zhao, Shigen Shen, and Changyuan Jiang. Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of networks*, 7(3):547, 2012.

[76] Amandeep Verma and Sakshi Kaushal. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Computing*, 62:1–19, 2017.

[77] Entisar S Alkayal, Nicholas R Jennings, and Maysoon F Abulkhair. Efficient task scheduling multi-objective particle swarm optimization in cloud computing. In *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*, pages 17–24. IEEE, 2016.

[78] Fadi Al-Turjman, Mohammed Zaki Hasan, and Hussain Al-Rizzo. Task scheduling in cloud-based survivability applications using swarm optimization in IoT. *Transactions on Emerging Telecommunications Technologies*, 30(8):e3539, 2019.

[79] Federico Marini and Beata Walczak. Particle swarm optimization (PSO). a tutorial. *Chemometrics and Intelligent Laboratory Systems*, 149:153–165, 2015.

[80] Jie Gao, Mitsuo Gen, Linyan Sun, and Xiaohui Zhao. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. *Computers & Industrial Engineering*, 53(1):149–162, 2007.

[81] Sung Ho Jang, Tae Young Kim, Jae Kwon Kim, and Jong Sik Lee. The study of genetic algorithm-based task scheduling for cloud computing. *International Journal of Control and Automation*, 5(4):157–162, 2012.

[82] Bahman Keshanchi, Alireza Souri, and Nima Jafari Navimipour. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. *Journal of Systems and Software*, 124:1–21, 2017.

[83] Henrique Yoshikazu Shishido, Júlio Cezar Estrella, Claudio Fabiano Motta Toledo, and Marcio Silva Arantes. Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds. *Computers & Electrical Engineering*, 69:378–394, 2018.

[84] Sunita Dhingra, Satinder Bal Gupta, and Ranjit Biswas. Genetic algorithm parameters optimization for bi-criteria multiprocessor task scheduling using design of experiments. *International Journal of Computer, Control, Quantum and Information Engineering*, 8(4):661–667, 2014.

[85] P Mathiyalagan, S Suriya, and SN Sivanandam. Modified ant colony algorithm for grid scheduling. *International Journal on computer science and Engineering*, 2(02):132–139, 2010.

[86] Arash Ghorbannia Delavar, Javad Bayrampoor, Ali Reza Khalili Boroujeni, and Ali Broumandnia. Task scheduling in grid environment with ant colony method for cost and time. *International Journal of Computer Science, Engineering and Applications*, 2(5):1, 2012.

[87] G Umarani Srikanth, V Uma Maheswari, P Shanthi, and Arul Siromoney. Tasks scheduling using ant colony optimization. 2012.

[88] Wei-Neng Chen and Jun Zhang. An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 39(1):29–43, 2008.

[89] Emetis Niazmand, Arash Ghorbannia Delavar, Javad Bayrampoor, and Ali Reza Khalili Boroujeni. JSWA: An improved algorithm for grid workflow scheduling using ant colony optimization. *J. Math. Comput. Sci.*, 6:315–331, 2013.

[90] Chengfeng Jian, Yekun Wang, Meng Tao, and Meiyu Zhang. Time-constrained workflow scheduling in cloud environment using simulation annealing algorithm. *Journal of Engineering Science & Technology Review*, 6(5), 2013.

[91] Min Dai, Dunbing Tang, Adriana Giret, Miguel A Salido, and Wei Dong Li. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robotics and Computer-Integrated Manufacturing*, 29(5):418–429, 2013.

[92] Ioannis A Moschakis and Helen D Karatza. Towards scheduling for internet-of-things applications on clouds: a simulated annealing approach. *Concurrency and Computation: Practice and Experience*, 27(8):1886–1899, 2015.

[93] Sayantani Basu, Marimuthu Karuppiah, K Selvakumar, Kuan-Ching Li, SK Hafizul Islam, Mohammad Mehedi Hassan, and Md Zakirul Alam Bhuiyan. An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment. *Future Generation Computer Systems*, 88:254–261, 2018.

[94] Djabir Abdeldjalil Chekired, Lyes Khoukhi, and Hussein T Mouftah. Industrial IoT data scheduling based on hierarchical fog computing: A key for enabling smart factory. *IEEE Transactions on Industrial Informatics*, 14(10):4590–4602, 2018.

[95] Georgios L Stavrinides and Helen D Karatza. A hybrid approach to scheduling real-time iot workflows in fog and cloud environments. *Multimedia Tools and Applications*, 78(17):24639–24655, 2019.

[96] Xiumin Zhou, Gongxuan Zhang, Jin Sun, Junlong Zhou, Tongquan Wei, and Shiyan Hu. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future Generation Computer Systems*, 93:278–289, 2019.

[97] Deze Zeng, Lin Gu, Song Guo, Zixue Cheng, and Shui Yu. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. *IEEE Transactions on Computers*, 65(12):3702–3712, 2016.

[98] Cristian Barca, Claudiu Barca, Cristian Cucu, Mariuca-Roxana Gavriloaia, Radu Vizireanu, Octavian Fratu, and Simona Halunga. A virtual cloud computing provider for mobile devices. In *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–4. IEEE, 2016.

[99] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-efficient scheduling heuristics for deadline constrained workloads on hybrid clouds. In *2011 IEEE third international conference on cloud computing technology and science*, pages 320–327. IEEE, 2011.

[100] Rodrigo N Calheiros and Rajkumar Buyya. Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds. In *International Conference on Web Information Systems Engineering*, pages 171–184. Springer, 2012.

[101] Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong, and Runqun Xiong. Bar: An efficient data locality driven task scheduling algorithm for cloud computing. In *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 295–304. IEEE, 2011.

[102] Ruilong Deng, Rongxing Lu, Chengzhe Lai, Tom H Luan, and Hao Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet of Things Journal*, 3(6):1171–1181, 2016.

[103] Enzo Baccarelli, Michele Scarpiniti, and Alireza Momenzadeh. EcoMobiFog-Design and Dynamic Optimization of a 5G Mobile-Fog-Cloud Multi-Tier Ecosystem for the Real-Time Distributed Execution of Stream Applications. *IEEE Access*, 7:55565–55608, 2019.

[104] Vitor Barbosa C Souza, Wilson Ramírez, Xavier Masip-Bruin, Eva Marín-Tordera, G Ren, and Ghazal Tashakor. Handling service allocation in combined fog-cloud scenarios. In *2016 IEEE international conference on communications (ICC)*, pages 1–5. IEEE, 2016.

[105] Vitor Barbosa Souza, Xavi Masip-Bruin, Eva Marín-Tordera, Wilson Ramírez, and Sergio Sanchez. Towards distributed service allocation in fog-to-cloud (F2C) scenarios. In *2016 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE, 2016.

[106] Olena Skarlat, Stefan Schulte, Michael Borkowski, and Philipp Leitner. Resource provisioning for IoT services in the fog. In *2016 IEEE 9th international conference on service-oriented computing and applications (SOCA)*, pages 32–39. IEEE, 2016.

[107] Wilson Ramírez, Xavier Masip-Bruin, Eva Marin-Tordera, Vitor Barbosa C Souza, Admela Jukan, Guang-Jie Ren, and O Gonzalez de Dios. Evaluating the benefits of combined and continuous Fog-to-Cloud architectures. *Computer Communications*, 113:43–52, 2017.

[108] Yucen Nan, Wei Li, Wei Bao, Flavia C Delicato, Paulo F Pires, and Albert Y Zomaya. Cost-effective processing for delay-sensitive applications in cloud of things systems. In *2016 IEEE 15th international symposium on network computing and applications (NCA)*, pages 162–169. IEEE, 2016.

[109] Mohammad Aazam and Eui-Nam Huh. Dynamic resource provisioning through fog micro datacenter. In *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, pages 105–110. IEEE, 2015.

[110] Haoyu Wang, Jiaqi Gong, Yan Zhuang, Haiying Shen, and John Lach. HealthEdge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 1213–1222. IEEE, 2017.

[111] Wenyuan Xu, Wade Trappe, Yanyong Zhang, and Timothy Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing*, pages 46–57, 2005.

[112] Guevara Noubir and Guolong Lin. Low-power DoS attacks in data wireless LANs and countermeasures. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):29–30, 2003.

[113] Liang Xiao, Larry J Greenstein, Narayan B Mandayam, and Wade Trappe. Channel-based detection of sybil attacks in wireless networks. *IEEE Transactions on Information Forensics and Security*, 4(3):492–503, 2009.

[114] Yingying Chen, Wade Trappe, and Richard P Martin. Detecting and localizing wireless spoofing attacks. In *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 193–202. IEEE, 2007.

[115] Seong Ho Chae, Wan Choi, Jung Hoon Lee, and Tony QS Quek. Enhanced secrecy in stochastic wireless networks: Artificial noise with secrecy protected zone. *IEEE Transactions on Information Forensics and Security*, 9(10):1617–1628, 2014.

[116] Y-W Peter Hong, Pang-Chang Lan, and C-C Jay Kuo. Enhancing physical-layer secrecy in multiantenna wireless systems: An overview of signal processing approaches. *IEEE Signal Processing Magazine*, 30(5):29–40, 2013.

[117] Tommaso Pecorella, Luca Brilli, and Lorenzo Mucchi. The role of physical layer security in IoT: A novel perspective. *Information*, 7(3):49, 2016.

[118] Maxwell Young and Raouf Boutaba. Overcoming adversaries in sensor networks: A survey of theoretical models and algorithmic approaches for tolerating malicious interference. *IEEE Communications Surveys & Tutorials*, 13(4):617–641, 2011.

[119] Tapolina Bhattasali and Rituparna Chaki. A survey of recent intrusion detection systems for wireless sensor network. In *International conference on network security and applications*, pages 268–280. Springer, 2011.

[120] Ajay Mahimkar and Theodore S Rappaport. SecureDAV: A secure data aggregation and verification protocol for sensor networks. In *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, volume 4, pages 2175–2179. IEEE, 2004.

[121] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265, 2003.

[122] Alberto Coen Porisini and Sabrina Sicari. SeDAP: Secure data aggregation protocol in privacy aware wireless sensor networks. In *International Conference on Sensor Systems and Software*, pages 135–150. Springer, 2010.

[123] Joao Girao, Dirk Westhoff, and Markus Schneider. CDA: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *IEEE International Conference on Communications, 2005. ICC 2005. 2005*, volume 5, pages 3044–3049. IEEE, 2005.

[124] Claude Castelluccia, Einar Mykletun, and Gene Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *The second annual international conference on mobile and ubiquitous systems: networking and services*, pages 109–117. IEEE, 2005.

[125] Roberto Riggio and Sabrina Sicari. Secure aggregation in hybrid mesh/sensor networks. In *2009 International Conference on Ultra Modern Telecommunications & Workshops*, pages 1–6. IEEE, 2009.

[126] Sabrina Sicari, Luigi Alfredo Grieco, Gennaro Boggia, and Alberto Coen-Porisini. DyDAP: A dynamic data aggregation scheme for privacy aware wireless sensor networks. *Journal of Systems and Software*, 85(1):152–166, 2012.

[127] A Boudguiga, A Olivereau, and N Oualha. Server assisted key establishment protocol for WSN: a MIKEY-ticket approach. *12th IEEE Trustcom*, 10, 2013.

[128] Shahid Raza, Simon Duquennoy, Tony Chung, Dogan Yazar, Thiemo Voigt, and Utz Roedig. Securing communication in 6LoWPAN with compressed IPsec. In *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pages 1–8. IEEE, 2011.

[129] Hassen Redwan Hussen, Gebere Akele Tizazu, Miao Ting, Taekkyeun Lee, Youngjun Choi, and Ki-Hyung Kim. SAKES: Secure authentication and key establishment scheme for M2M communication in the IP-based wireless sensor network (6L0WPAN). In *2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 246–251. IEEE, 2013.

[130] Yosra Ben Saied and Alexis Olivereau. HIP Tiny Exchange (TEX): A distributed key exchange scheme for HIP-based Internet of Things. In *Third International Conference on Communications and Networking*, pages 1–8. IEEE, 2012.

[131] Wenliang Du, Jing Deng, Yunghsiang S Han, and Pramod K Varshney. A key predistribution scheme for sensor networks using deployment knowledge. *IEEE Transactions on dependable and secure computing*, 3(1):62–77, 2006.

[132] Piotr Szczechowiak and Martin Collier. Tinyibe: Identity-based encryption for heterogeneous sensor networks. In *2009 International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pages 319–354. IEEE, 2009.

[133] Giacomo De Meulenaer, François Gosset, François-Xavier Standaert, and Olivier Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, pages 580–585. IEEE, 2008.

[134] René Hummen, Jan H Ziegeldorf, Hossein Shafagh, Shahid Raza, and Klaus Wehrle. Towards viable certificate-based authentication for the internet of things. In *Proceedings of the 2nd ACM workshop on Hot topics on wireless network security and privacy*, pages 37–42. ACM, 2013.

[135] R Cragie, Y Ohba, R Moskowitz, Z Cao, and B Sarikaya. Security bootstrapping solution for resource-constrained devices. *IETF (work in progress) Available [Online] http://tools. ietf. org/html/draft-oflynn-corebootstrapping-03*, 2010.

[136] Lijun Yang, Chao Ding, and Meng Wu. Establishing authenticated pairwise key using pairing-based cryptography for sensor networks. In *2013 8th International Conference on Communications and Networking in China (CHINACOM)*, pages 517–522. IEEE, 2013.

[137] Rodrigo Roman, Cristina Alcaraz, Javier Lopez, and Nicolas Sklavos. Key management systems for sensor networks in the context of the internet of things. *Computers & Electrical Engineering*, 37(2):147–159, 2011.

[138] Chung Kei Wong, Mohamed Gouda, and Simon S Lam. Secure group communications using key graphs. *IEEE/ACM transactions on networking*, 8(1):16–30, 2000.

[139] Bob Briscoe. MARKS: Zero side effect multicast key management using arbitrarily revealed key sequences. In *International Workshop on Networked Group Communication*, pages 301–320. Springer, 1999.

[140] Luca Veltri, Simone Cirani, Stefano Busanelli, and Gianluigi Ferrari. A novel batch-based group key management protocol applied to the internet of things. *Ad Hoc Networks*, 11(8):2724–2737, 2013.

[141] Mohammed Riyadh Abdmeziem, Djamel Tandjaoui, and Imed Romdhani. A decentralized batch-based group key management protocol for mobile internet of things (DBGK). In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1109–1117. IEEE, 2015.

[142] Christof Paar and Jan Pelzl. The advanced encryption standard (aes). In *Understanding cryptography*, pages 87–121. Springer, 2010.

[143] Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. New lightweight des variants. In *International Workshop on Fast Software Encryption*, pages 196–210. Springer, 2007.

[144] Shahid Raza, Thiemo Voigt, and Vilhelm Jutvik. Lightweight IKEv2: a key management solution for both the compressed ipsec and the ieee 802.15. 4 security. In *Proceedings of the IETF workshop on smart object security*, volume 23. Citeseer, 2012.

[145] Robert Moskowitz, Pekka Nikander, Petri Jokela, and Thomas Henderson. Host identity protocol. Technical report, RFC 5201, April, 2008.

[146] Robert Moskowitz, Tobias Heer, Petri Jokela, and T Henderson. Host identity protocol version 2 (HIPv2). *RFC7401, Updated by RFC8002, IETF*, 2015.

[147] Robert Moskowitz and R Hummen. Hip diet exchange (DEX). *draft-moskowitz-hip-dex-00 (WiP), IETF*, 2012.

[148] Bernard Aboba, Larry Blunk, John Vollbrecht, James Carlson, Henrik Levkowetz, et al. Extensible authentication protocol (EAP). 2004.

[149] Tom Phelan. Datagram transport layer security (DTLS) over the datagram congestion control protocol (DCCP). Technical report, RFC 5238, May, 2008.

[150] Frank Alexander Kraemer, Anders Eivind Braten, Nattachart Tamkittikhun, and David Palma. Fog computing in healthcare—a review and discussion. *IEEE Access*, 5:9206–9222, 2017.

[151] Muhammad Kafil and Ishfaq Ahmad. Optimal task assignment in heterogeneous distributed computing systems. *IEEE concurrency*, 6(3):42–50, 1998.

[152] Stephen A Cook. An overview of computational complexity. *Communications of the ACM*, 26(6):400–408, 1983.

[153] Evangelos Triantaphyllou. Multi-criteria decision making methods. In *Multi-criteria decision making methods: A comparative study*, pages 5–21. Springer, 2000.

[154] Randa M Abdelmoneem, Abderrahim Benslimane, Eman Shaaban, Sherin Abdelhamid, and Salma Ghoneim. A cloud-fog based architecture for IoT applications dedicated to healthcare. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2019.

[155] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K Ghosh, and Rajkumar Buyya. iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments. *Software: Practice and Experience*, 47(9):1275–1296, 2017.

[156] Meriem Kassar, Brigitte Kervella, and Guy Pujolle. An overview of vertical handover decision strategies in heterogeneous wireless networks. *Computer Communications*, 31(10):2607–2620, 2008.

[157] Ahmed H Zahran, Ben Liang, and Aladdin Saleh. Signal threshold adaptation for vertical handoff in heterogeneous wireless networks. *Mobile Networks and Applications*, 11(4):625–640, 2006.

[158] I El Fachtali, R Saadane, and M Koutbi. A survey of handovers decision algorithms for next generation wireless networks. *network*, 4(1), 2015.

[159] Abhijit Sarma, Sandip Chakraborty, and Sukumar Nandi. Deciding handover points based on context-aware load balancing in a WiFi-WiMAX heterogeneous network environment. *IEEE Transactions on Vehicular Technology*, 65(1):348–357, 2016.

[160] Gregory P Pollini. Trends in handover design. *IEEE Communications magazine*, 34(3):82–90, 1996.

[161] Enrique Stevens-Navarro, Yuxia Lin, and Vincent WS Wong. An MDP-based vertical handoff decision algorithm for heterogeneous wireless networks. *IEEE Transactions on Vehicular Technology*, 57(2):1243–1254, 2008.

[162] Qian Zhang, Chuanxiong Guo, Zihua Guo, and Wenwu Zhu. Efficient mobility management for vertical handoff between WWAN and WLAN. *IEEE Communications Magazine*, 41(11):102–108, 2003.

[163] Wei Bao, Dong Yuan, Zhengjie Yang, Shen Wang, Wei Li, Bing Bing Zhou, and Albert Y Zomaya. Follow me fog: toward seamless handover timing schemes in a fog computing environment. *IEEE Communications Magazine*, 55(11):72–78, 2017.

[164] Ick-Jae Yoon, Hyungrak Kim, and Young Joong Yoon. UWB RF receiver front-end with band-notch characteristic of 5 GHz WLAN. In *2006 IEEE Antennas and Propagation Society International Symposium*, pages 1303–1306. IEEE, 2006.

[165] Bart Braem, Benoît Latré, Chris Blondia, Ingrid Moerman, and Piet Demeester. Improving reliability in multi-hop body sensor networks. In *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*, pages 342–347. IEEE, 2008.

[166] Wenhui Zhang, Juergen Jaehnert, and Klaus Dolzer. Design and evaluation of a handover decision strategy for 4th generation mobile networks. In *The 57th IEEE Semiannual Vehicular Technology Conference, 2003. VTC 2003-Spring.*, volume 3, pages 1969–1973. IEEE, 2003.

[167] K-I Itoh, Soichi Watanabe, J-S Shih, and Takuro Sato. Performance of handoff algorithm based on distance and RSSI measurements. *IEEE transactions on vehicular technology*, 51(6):1460–1468, 2002.

[168] Atreyi Bose and Chuan Heng Foh. A practical path loss model for indoor WiFi positioning enhancement. In *2007 6th International Conference on Information, Communications & Signal Processing*, pages 1–5. IEEE, 2007.

[169] Ehsan Ahvar, Anne-Cécile Orgerie, and Adrien Lebre. Estimating energy consumption of cloud, fog and edge computing infrastructures. *IEEE Transactions on Sustainable Computing*, 2019.

[170] Daniel Halperin, Ben Greenstein, Anmol Sheth, and David Wetherall. Demystifying 802.11 n power consumption. In *Proceedings of the 2010 international conference on Power aware computing and systems*, page 1, 2010.

[171] Seungmoon Song and Hartmut Geyer. Regulating speed and generating large speed transitions in a neuromuscular human walking model. In *2012 IEEE International Conference on Robotics and Automation*, pages 511–516. IEEE, 2012.

[172] R McNeill Alexander. Energetics and optimization of human walking and running: the 2000 raymond pearl memorial lecture. *American journal of human biology*, 14(5):641–648, 2002.

[173] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[174] Yunbo Li, Anne-Cécile Orgerie, Ivan Rodero, Betsegaw Lemma Amersho, Manish Parashar, and Jean-Marc Menaud. End-to-end energy models for Edge Cloud-based IoT platforms: Ap-

plication to data stream analysis in IoT. *Future Generation Computer Systems*, 87:667–678, 2018.

[175] Ramia Babiker Mohammed Abdelrahman, Amin Babiker A Mustafa, and Ashraf A Osman. A comparison between IEEE 802.11 n and ac standards. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 17(5):30–34, 2015.

[176] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems*, 13(3):260–274, 2002.

[177] Tiffanie Ramos, Sandra Dedesko, Jeffrey A Siegel, Jack A Gilbert, and Brent Stephens. Spatial and temporal variations in indoor environmental conditions, human occupancy, and operational characteristics in a new hospital building. *PLoS One*, 10(3): e0118207. https://doi.org/10.1371/journal.pone.0118207(3), 2015.

[178] A Lagar Cavilla, Gerard Baron, Thomas E Hart, Lionel Litty, and Eyal De Lara. Simplified simulation models for indoor MANET evaluation are not robust. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, pages 610–620. IEEE, 2004.

[179] CISCO Meraki. *High Density Wi-Fi Deployments*, (accessed April 12, 2020). `https://documentation.meraki.com`.

[180] Berkeley Software Distribution. *Intel Edison*, (accessed November 3, 2020). `https://documentation.meraki.com`.

[181] Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018.

[182] Tuan Nguyen Gia, Mingzhe Jiang, Amir-Mohammad Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. 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*, pages 356–363. IEEE, 2015.

[183] Shahid Raza, Hossein Shafagh, Kasun Hewage, René Hummen, and Thiemo Voigt. Lithe: Lightweight secure CoAP for the internet of things. *IEEE Sensors Journal*, 13(10):3711–3720, 2013.

[184] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. A DTLS based end-to-end security architecture for the Internet of Things with two-way authentication. In *37th Annual IEEE Conference on Local Computer Networks-Workshops*, pages 956–963. IEEE, 2012.

[185] Thomas Kothmayr, Corinna Schmitt, Wen Hu, Michael Brünig, and Georg Carle. DTLS based security and two-way authentication for the Internet of Things. *Ad Hoc Networks*, 11(8):2710–2723, 2013.

[186] René Hummen, Hossein Shafagh, Shahid Raza, Thiemo Voig, and Klaus Wehrle. Delegation-based authentication and authorization for the IP-based internet of things. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 284–292. Ieee, 2014.

[187] Jorge Granjal, Edmundo Monteiro, and Jorge Sa Silva. End-to-end transport-layer security for Internet-integrated sensing applications with mutual and delegated ECC public-key authentication. In *2013 IFIP Networking Conference*, pages 1–9. IEEE, 2013.

[188] Shahid Raza, Ludwig Seitz, Denis Sitenkov, and Göran Selander. S3K: Scalable security with symmetric keys—DTLS key establishment for the Internet of Things. *IEEE Transactions on Automation Science and Engineering*, 13(3):1270–1280, 2016.

[189] Sanaz Rahimi Moosavi, Tuan Nguyen Gia, Ethiopia Nigussie, Amir M Rahmani, Seppo Virtanen, Hannu Tenhunen, and Jouni Isoaho. End-to-end security scheme for mobility enabled healthcare internet of things. *Future Generation Computer Systems*, 64:108–124, 2016.

[190] Priyan Malarvizhi Kumar and Usha Devi Gandhi. Enhanced DTLS with CoAP-based authentication scheme for the internet of things in healthcare application. *The Journal of Supercomputing*, pages 1–21, 2017.

[191] Arago Systems. Wismote. `http://www.aragosystems.com/produits/wisnet/wismote`, 2020.

[192] Texas Instruments. *CC2520 DATASHEET 2.4 GHZ IEEE 802.15.4/ZIGBEE® RF TRANSCEIVER*, 12 2007. Rev. 1.

[193] Olaf Bergmann. *TinyDTLS*, (accessed November 3, 2020). `https://sourceforge.net/projects/tinydtls/`.

[194] Aaron Gifford. *SHA*, (accessed November 3, 2020). `https://aarongifford.com/computers/sha.html`.