Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

Abbes Laghrour University -Khenchela

Faculty of Sciences and Technology

Department of Mathematics and Computer Science

N° Order:

Serie:

ARTIFICIAL INTELLIGENCE BASED APPROACH FOR IOT SERVICES COMPOSITION BY ANALYZING THE USER NEEDS

Thesis presented by

Rabah BOUCETTI

To obtain the degree of

DOCTOR IN COMPUTER SCIENCE

(Option: Artificial Intelligence)

Directed by: **Dr. Sofiane Mounine HAMEM**
Pr. Ouassila HIOUAL

Presented publically in: ___ / ___ / ____

In front of the jury, composed:

N°	First & Family name	Grade	Establishment	Quality
01	Jamel NESSAH	MCA	Abbes Laghrour University –Khenchela	President
02	Sofiane Mounine HEMAM	MCA	Abbes Laghrour University –Khenchela	Supervisor
03	Ouassila HIOUAL	Prof	Abbes Laghrour University –Khenchela	Co-Supervisor
04	Zianou AHMED SEGHIR	MCA	Abbes Laghrour University –Khenchela	Examiner
05	Mohamed AMROUNE	Prof	Larbi Tebessi University –Tebessa	Examiner
06	Manel KOLI	MCA	National Higher School, Assia Djebar – Constantine	Examiner

[2022-2023]

Acknowledgments

Before inviting you to the presentation of this work, I have the opportunity to express my gratitude to all the people who, through their help and encouragement, have enabled me to complete this dissertation.

First of all, I would like to thank Dr. Sofiane Mounine HAMEM and Prof. Ouassila HIOUAL for the trust they have kindly granted me, for their precious help, their advice, as well as their availability.

My thanks go to the honorable president of the jury for agreeing to examine my work.

May the members of this prestigious and distinguished jury be assured of my gratitude for having done me the honor of evaluating my work.

Dedication

To the pure soul of my father,

To my dear mother, may Allah protect her,

To my very dear wife in whom I base all my hopes,

To my little angels, Razane & Rafi

To all my family members,

To all those who have participated directly or indirectly in the realization of this work.

In testimony of all my affection.

Abstract

In the Internet of Things (IoT) environment, the services provided by the connected objects are published as services through the web. This allows to machines to interact between them and, makes the IoT services composition possible. However, the vast proliferation of smart object generates services with the same functionalities but different in terms of quality of services (QoS) proprieties. This makes the satisfaction of the user requirements often complex and a NP-hard problem. Indeed, respecting the QoS constraints (user preferences in terms of QoS) is a challenge, due to the high number of candidate services for the composition. This challenge consists of selecting the most appropriate services so that the composite service must meet both the functional and the non-functional requirements of the user. To deal with this challenge, we propose an approach based on Genetic Algorithms (GA) and Neural Networks (NN) for QoS-aware IoT Services Composition in the context of large-scale environments. The combination between GA and NN allows finding the quasi-optimal IoT service composition. This latter is based on global QoS optimization. To reach this objective, the QoS intervals are decomposed into M QoS-levels to engage them into the theoretical composition. Then, the proposed first GA is used to obtain the ideal theoretical composition with an overall QoS optimization. Afterward, the proposed NN is used to eliminate the inappropriate concrete IoT services, and retain only the services having the same categories of the atomic theoretical services composing the ideal theoretical composition. This allows us to optimize the search space as well as the execution time. Finally, we apply the second GA on the concrete services of the retained categories, in order to obtain the IoT service concrete composition with an overall QoS optimization. The simulation results show that the proposed approach has the best composition time, the best Hypervolume indicator and the best compositional optimality compared to SC-FLA, Improved GA and MGA approaches. On another side, it has almost the same performances compared to TS-QCA and, it finds the near-optimal composition in a very short time compared to PSA, which is an optimal approach. Thus, the obtained results show the effectiveness of our approach.

Keywords: IoT Service, IoT Service Composition, Quality of Service decomposition, QoS-aware service composition, Neural Network, Genetic Algorithm

Résumé

Dans l'environnement de l'Internet des Objets (IdO), les services fournis par les objets connectés sont publiés en tant que services via le Web. Cela permet aux machines d'interagir entre elles et rend possible la composition des services IoT. Cependant, la vaste prolifération d'objets intelligents génère des services avec les mêmes fonctionnalités mais différents en termes de propriétés de qualité de services (QoS). Cela rend la satisfaction des besoins des utilisateurs souvent complexe et un problème NP-difficile. En effet, le respect des contraintes de QoS (préférences des utilisateurs en termes de QoS) est un challenge, du fait du nombre élevé de services candidats à la composition. Cet enjeu consiste à sélectionner les services les plus appropriés pour que le service composite réponde à la fois aux exigences fonctionnelles et non fonctionnelles de l'utilisateur. Pour relever ce défi, nous proposons une approche basée sur les algorithmes génétiques (GA) et les réseaux de neurones (NN) pour la composition des services IoT sensible à la QoS dans le contexte d'environnements à grande échelle. La combinaison entre GA et NN permet de trouver la composition de service IoT quasi-optimale. Cette dernière est basée sur une optimisation globale de la QoS. Pour atteindre cet objectif, les intervalles de QoS sont décomposés en M niveaux de QoS pour les engager dans la composition théorique. Ensuite, le premier GA proposé est utilisé pour obtenir la composition théorique idéale avec une optimisation globale de la QoS. Par la suite, le NN proposé est utilisé pour éliminer les services IoT concrets inappropriés, et ne retenir que les services ayant les mêmes catégories de services théoriques atomiques composant la composition théorique idéale. Cela nous permet d'optimiser l'espace de composition ainsi que le temps d'exécution. Enfin, nous appliquons le deuxième GA sur les services concrets des catégories retenues, afin d'obtenir la composition concrète du service IoT avec une optimisation globale de la QoS. Les résultats de la simulation montrent que l'approche proposée a le meilleur temps de composition, le meilleur indicateur d'hypervolume et la meilleure optimalité de composition par rapport aux approches SC-FLA, GA améliorée et MGA. D'un autre côté, il a presque les mêmes performances par rapport au TS-QCA et, il retrouve la composition quasi-optimale en très peu de temps par rapport au PSA, qui est une approche optimale. Ainsi, les résultats obtenus montrent l'efficacité de notre approche.

Mots clés : Service IoT, composition de services IoT, décomposition de la qualité de service, Composition du service sensible à QoS, Réseau de neurones, Algorithme génétique.

الملخص

في بيئة إنترنت الأشياء (IoT) ،يتم نشر الخدمات التي توفرها الكائنات الذكية المتصلة بالأنترنت كخدمات عبر الويب. مما يتيح لهذه الاشياء بالتفاعل فيما بينها، ويجعل تركيب خدماتها ممكنًا. ومع ذلك، فإن الانتشار الواسع للأشياء الذكية يولد خدمات لها نفس الوظائف ولكنها تختلف من حيث خصائص جودة الخدمات (QoS). هذا ما يجعل من تلبية متطلبات المستخدم مشكلة صعبة ومعقدة في كثير من الأحيان. في الواقع، يعد احترام القيود المفروضة على جودة الخدمات (تفضيلات المستخدم) تحديًا، نظرًا للعدد الكبير من الخدمات المرشحة لعملية التركيب. يتجلى هذا التحدي في اختيار أنسب الخدمات بحيث تفي الخدمة المركبة بالمتطلبات الوظيفية وغير الوظيفية للمستخدم. للتعامل مع هذا التحدي، نقترح مقاربة تعتمد على الخوارزميات الجينية (خ.ج) والشبكات العصبية (ش.ع) لتكوين الخدمة المركبة مع الاخذ بعين الاعتبار جودة الخدمات. يتيح الجمع بين خ.ج و ش.ع الحصول على خدمة إنترنت الأشياء المركبة الشبه الأمثل وبجودة خدمات شاملة محسنة. للوصول إلى هذا الهدف، نقسم مجالات جودة الخدمات إلى مستويات نظرية، لإشراكهم في تركيب الخدمة النظرية. من اجل ذلك، يتم استخدام خ.ج للحصول على التركيب النظري المثالي. ثم، باستخدام ش.ع المقترحة، نتخلص من خدمات إنترنت الأشياء غير الملائمة، ونحتفظ فقط بالخدمات التي لها نفس فئات الخدمات النظرية العضوية التي تشكل الخدمة المركبة النظرية المثالية. هذا يسمح لنا بتقليص فضاء العمل وكذا وقت التركيب. أخيرًا، نطبق خ.ج مرة أخرى على خدمات إنترنت الأشياء الحقيقية المحتفظ بها، من أجل الحصول على خدمة مركبة حقيقية مع تحسين شامل لجودة الخدمات. تظهر نتائج التجارب أن المقاربة المقترحة توفر أفضل وقت لتركيب الخدمات، وأفضل مؤشر Hypervolume ، وكذا أفضل تركيبة مثالية مقارنة بمقاربات أخرى (SC-FLA, Improved GA and MGA). من ناحية أخرى، للمقاربة المقترحة نفس الأداء تقريبًا مقارنة بـ TS-QCA، ونحصل على التركيب شبه الأمثل في وقت قصير جدًا مقارنة بـ PSA، وهي مقاربة مقترحة للحصول على التركيب المثالي. النتائج المتحصل عليها تظهر فعالية المقاربة المقترحة.

الكلمات المفتاحية: خدمة إنترنت الأشياء، تركيب خدمات إنترنت الأشياء، تقسيم مجال جودة الخدمات، تركيب الخدمات المرتبط بجودة الخدمات، الشبكة العصبية ، الخوارزمية الجينية.

Contents

List of Figures

List of Tables

Listings

Acronyms

1G *First Generation of wireless cellular technology*

2G *Second-Generation mobile network*

3G *third Generation of cellular technology*

4G LTE *Fourth Generation Long-Term Evolution*

5G *Fifth-Generation of mobile telecommunications technology*

6LoWPAN *IPv6 Low Power Wireless Personal Area Networks*

ABS *Antilock Braking System*

ACS *Abstract Composite Services*

AGC *Apollo Guidance Computer*

AI *Artificial Intelligence*

AmI *Ambient Intelligence*

AMQP *Advanced Message Queuing Protocol*

API *Application Programming Interface*

ARM *Advanced RISC Machines*

AS *Abstract Service*

ASIP *Application-Specific Instruction-set Processor*

BLE *Bluetooth Low Energy*

Bluetooth SIG *Bluetooth Special Interest Group*

BSD *Berkeley Source Distribution*

CD *Compact Disc*

CoAP *Constrained Application Protocol*

CoRE *Constrained RESTful Environments*

CPU *Central Processing Unit*

CQCA *Clustering-based and QoS-aware service Composition Algorithm*

CS *Concrete Service*

CSV *Comma Separated Values*

DSP *Digital Signal Processing*

EQSA *Energy-centric and QoS-aware IoT services Selection Algorithm*

FDD *Frequency Division Duplex*

FTP *File Transfer Protocol*

GA *Genetic Algorithm*

GNN-QSC *Genetic Algorithms and Neural Networks for QoS-aware IoT Services Composition*

GPS *Global Positioning System*

GPWS *Ground Proximity Warning System*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hypertext Transfer Protocol Secure*

HV *hypervolume*

IDE *Integrated Development Environment*

IEEE *Institute of Electrical and Electronics Engineers*

IIoT *Industrial Internet of Things*

IoT *Internet of Things*

IP *Internet Protocol*

IPv4 *Internet Protocol version 4*

IPv6 *Internet Protocol version 6*

ISM *Industrial, Scientific, Medical*

ITC *Ideal Theoretical Composition*

JMS *Java Message Service*

JSON *JavaScript Object Notation*

JSON-LD *JavaScript Object Notation for Linked Data*

LAN *Local Area Network*

LoRa *Long Range*

LoRaWAN *Long Range Wide Area Network*

LPWAN *Low-Power Wide-Area Network*

M2M *Machine-to-Machine*

MAC *Media Access Control*

MADM *Multi-Attribute Decision-Making*

MCE *Multi-Cloud Environment*

MF *Matrix Factorization*

MGA *Multi-population Genetic Algorithm*

MIP *Mixed Integer Programming*

MMU *Memory Management Unit*

MOM *Message-Oriented Middleware*

MQTT *Message Queuing Telemetry Transport*

NB-IoT *Narrow-band IoT*

NFC *Near-Field Communication*

NSGA *Non-dominated Sorting Genetic Algorithm*

OS *Operating System*

OSI *Open Systems Interconnection*

PAN *Personal Area Network*

PC *Personal Computer*

PDA *Personal Digital Assistants*

PLA *Programmable Logic Array*

POP *Post Office Protocol*

PSA *Pareto-based partial Services selection Approach*

PSO *Particle Swarm Optimization*

QoS *Quality of Service*

RAM *Random Access Memory*

RDF *Ressources Description Framework*

REST *REpresentational State Transfer*

RFI *Radio Frequency Interface*

RFID *Radio Frequency Identification*

RISC *Reduced Instruction Set Computing*

ROM *Read Only Memory*

RTOS *Real-Time Operating System*

SAW *Simple Additive Weighting*

SFLA *Shuffled Frog Leaping Algorithm*

SIG *Special Interest Group*

SMACS *Service Monitoring And Composition System*

SMTP *Simple Mail Transfer Protocol*

SOA *Service Oriented Architecture*

SOAP *Simple Object Access Protocol*

SPARQL *Simple Protocol And RDF Query Language*

SWoT *Semantic Web of Things*

TAS *Theoretical Abstract Service*

TCP *Transmission Control Protocol*

TCP/IP *Transmission Control Protocol/Internet Protocol*

TD *Thing Description*

TDD *Time Division Duplex*

TS *Theoretical Service*

TS-QCA *Two-Step QoS service Composition Algorithm*

TS-QCA *Two-Steps QoS-aware services Composition Algorithm*

UDDI *Universal Description, Discovery, and Integration*

UDP *User Datagram Protocol*

UNB *Ultra Narrow Band*

URI *Uniform Resource Identifier*

USB *Universal Serial Bus*

W3C *World Wide Web Consortium*

WAN *Wide Area Network*

Wi-Fi *Wireless Fidelity*

WinCE *Windows Embedded Compact*

WLAN *Wireless Local Area Network*

WoT *Web of Things*

WPAN *Wireless Personal Area Networks*

WSDL *Web Services Description Language*

WSN *Wireless Sensor Networks*

XML *eXtensible Markup Language*

XMPP *Extensible Messaging and Presence Protocol*

Introduction

.1 Context

The *Internet of Things* (IoT) is a technical embodiment of ubiquitous computing as imagined in 1991 by Mark Weiser, where technology gradually disappears from the environment of users, naturally integrated inside everyday objects: telephones, watches, household appliances, etc [1]. These physical objects embed a growing number of sensors and actuators allowing them to communicate with each other, measure the environment and act on it. Very promising, this concept opens the way to a multitude of scenarios based on the interconnection between the physical world and the virtual world through the Internet. These different scenarios will have a huge impact on several aspects of daily life and will influence the behavior of both personal and business users. Home automation, daily assistance for dependent people (assisted living), health (e-health) and education are just a few examples in which the Internet of Things will play a very important role in the near future. As far as the business world is concerned, automation, industrial manufacturing, logistics, business management, and transport will be very affected by this new concept.

However, the implementation of IoT applications raises significant challenges: scaling up, interoperability between heterogeneous hardware and software technologies, distribution of software components on low-resource platforms, quality of service management to the dynamics of the environment to which are added security problems and respect for the privacy of users. The ubiquitous service-oriented computing paradigm provides a framework to address the majority of the challenges outlined above. Moreover, reuse and interoperability are at the heart of the IoT service paradigm. This technology enables seamless service reuse and interoperability that facilitates rapid development and integration of IoT applications. Thus the use of message-oriented inter-protocol gateways conveyed through the Web and weak coupling in Service Oriented Architectures (SOA) makes it possible to respond effectively to the problem of interoperability management, transition to scaling, and managing the distribution of software components. In addition, services composition, one of the most important concepts of the service-oriented paradigm, enables the provision of elaborate services that adapt to the user's needs and the dynamics of the environment.

From the perspective of the Service-Oriented Architecture (SOA), the implementation of IoT applications by composition of services can be considered as the collaboration of several services generally organized according to two modes: the centralized mode called the *orchestration* and distributed mode called *choreography* [2]. Orchestration requires the presence of a central entity which is responsible for controlling the interactions between the component services. This entity must have a complete view of the component services state and the complete plan for invoking these services. Considering that the availability of this central entity cannot be guaranteed in a highly dynamic IoT environment, the choreography mode is more appropriate in this case. In the latter, the component services have local intelligence allowing them to interact with other services so that this collective execution can achieve the service expected by the user. The interactions, in a choreography, are made directly (Peer to Peer) without going through a central entity, which allows better exploitation of network resources avoiding overloading of links and over-exploitation of nodes leading to the central one, and therefore the rapid exhaustion of the devices.

The use of IoT services, whether in orchestration or choreography, requires a preliminary stage of discovery [3], which consists of searching and locating the IoT services corresponding to the user's needs which are expressed by means of a request defining functional and non-functional, also called *Quality of Service* (QoS) criteria, expectations. Therefore, focusing on the problem of service selection being the main problem to be solved in the services discovery and composition. The services selection consists, in general, in choosing a service or a combination of concrete services among a set of candidates having substantially similar functionalities and capable of carrying out a required task. This choice is made on the basis of non-functional criteria in order to meet the non-functional users requirements. This problem is known by the "QoS-aware services composition" [4, 5, 6, 7, 8, 9, 10].

The selection of services must also take into account the constraints of the users in terms of QoS. These are generally expressed in a global and quantitative way using metrics such as response time, cost, reliability, etc. The user can require that the response time and the price of the composite service should not exceed 20 s and 50 Dollars respectively. This implies being able to estimate the overall quality of the composite service on each QoS criterion from the values of their components. The problem being in an IoT environment composed of thousands of objects offering quite similar functionalities in description but varying in terms of QoS values, the search for service composition solutions leads to exploring and evaluating a considerable number of solutions. In such a context, the selection of the best services entering the composition among sets of substantially equivalent services and each having multiple QoS constraints, is an NP-hard optimization problem. Moreover, in a composition under constraints, the selection must be made during the composition process so that the composition offers a better overall QoS.

Given the large number of IoT services available on the web, it is difficult, if not impossible, to compose them manually, which is why the composition must be automatic. In fact, the automatic composition allows an adaptation to the requirements of the users. It aims to make the most of the intrinsic properties of the IoT service platform. It is in this context that our work takes place, where we propose an automatic composition approach for IoT services based on *Artificial Intelligence* (AI) techniques and a multi-criteria

optimization strategy. The AI techniques are often considered the most effective and even predominant for automatically solving services composition problems [11]. Since the QoS-aware IoT services composition is an NP-hard problem in a large-scale environment, the objective is to find a composition near the optimum in a reasonable time while satisfying the non-functional user constraints and ensuring a better overall quality of service.

.2 Problem and motivations

Given a set of non-functional user preferences, the challenge is how to build a composite service in an automatic way, while respecting these preferences, and such that the overall value of QoS is optimal. In the state of the art, we found a large number of works on the development of algorithms for optimizing the composition of IoT services. They can generally be classified into one of the following classes : composition approaches based on functional properties, and composition approaches based on noun-functional (QoS) properties [12].

Our literature review revealed that first-class approaches attempt to find valid composition without considering the quality of service (QoS) of the solution sought. Therefore, these methods would be unable to find optimal compositions satisfying the user's constraints in terms of QoS. As for the QoS-aware IoT service composition approaches, these are approaches that can lead to optimal or near-to-optimal solutions in a reasonable time, while respecting the user's constraints. The main idea is to model the composition as an abstract workflow, i.e. assumes the prior existence of an abstract services composition plan, and each abstract service will subsequently be linked to a concrete service taken from among the concrete services that are functionally equivalent. However, the proposed approaches generally consisted of decomposing the global constraints into local constraints so that the selection could be done locally. This has given rise to over-constrained local sub-problems that often rule out good solutions.

In this thesis, we propose an approach for automatic composition of IoT services that considers non-functional properties (QoS parameters such as response time, cost, reputation, etc.) and non-functional user preferences (constraints imposed on these QoS parameters). For this, we have adopted an approach based on AI techniques, mainly Neural Networks and the Genetic Algorithm. The neural network will be used as a clustering technique to eliminate inappropriate services after their regrouping into clusters according to their QoS parameters. As for the genetic algorithm, it will be used to find the near-to-optimal composition in terms of global QoS, while respecting the imposed QoS constraints.

.3 Contributions

This work aims to promote the integration of user preferences in terms of QoS in the process of IoT service composition. Our contribution to solving the "QoS-aware IoT services automatic composition" problem is based on: neural networks to reduce the composition space; and the genetic algorithm to search for the quasi-optimal composition.

The user specifies non-functional constraints, i.e his preferences in terms of QoS. Thus, our method is able to find a composite service that optimizes the overall QoS value, while satisfying the specified constraints. We summarize our main contributions below:

1. **State of the art development of composition approaches:** we have carried out detailed state-of-the-art studies on QoS-aware IoT services composition approaches. A comparative analysis of the different contributions according to the evaluation criteria that we have defined has enabled us on the one hand, to identify the advantages and limits of each approach, and on the other hand, to better position our approach to see how it would be possible to improve this area of research.

2. **QoS-Intervals breakdown mechanism:** we decompose each QoS attribute, for each abstract service (AS_i), into M levels of QoS from the minimum value to the maximum value of each QoS attribute. Thus, we will obtain M theoretical services (M theoretical QoS vectors) for each abstract service. To generate a reduced number of the theoretical services used to find the Ideal Theoretical Composite ITC service. The latter makes it possible to give as a result, for each abstract service, the category of concrete services to be engaged in the concrete composition.

3. **Services (concrete and theoretical) evaluation mechanism:** we propose a neural network model to evaluate IoT and theoretical services. Then, group them according to the appropriate QoS level. The number of neurons in the input layer corresponds to the QoS attributes vector dimension.

4. **Find the near optimal composition:** for this we apply the adapted genetic algorithm on a reduced population after elimination of the inappropriate clusters of concrete services. The resulting composite service will have an optimal QoS while respecting the imposed QoS preferences.

5. **Validation and experiments:** we present the architecture of the proposed composition environment by detailing the different modules constituting it through an illustrative scenario. We also propose a set of experiments to evaluate and validate our contribution.

.4 Organization of the thesis

The chapters of this thesis are organized according to a logical structure starting with a general study of the different concepts of embedded systems as main components of the Internet of Things. Afterwards, the second chapter is devoted to the IoT paradigm where we will focus on the composition of IoT services. It is in the third chapter that we will develop the state of the art of QoS-aware IoT services composition, whose approaches based on AI techniques are present. The fourth chapter is thus devoted to the design of our approach, also based on AI. And finally, the last chapter will be devoted to the evaluation and validation of our proposal. At the end, a general conclusion to draw up the results and prospects.

- **Chapter 1: Embedded systems generalities**

  In this chapter we will make a non-exhaustive detour on the main axes along which research and industry are articulated the field of embedded systems.We will first discuss some theoretical definitions of the subject, then consider one of the most predominant classifications in the field, to finish with an overview of the main characteristics that govern embedded systems.

- **Chapter 2: IoT: Theoretical bases, principles and technology**

  This chapter will devoted the IoT technologies. The IoT is a relatively new concept. It is a growing field, and its development requires the support of some innovative technologies. In this chapter, we present the concepts, the theoretical bases, the principles, the characteristics and the different technologies involved in the IoT.

- **Chapter 3: IoT services composition, State of the art**

  This chapter has two parts: in the first part, we present the fundamental concepts of IoT service and its properties by emphasizing the non-functional aspect, in particular Quality of Service (QoS) properties. Then, we present the services composition process and its steps, of which the service selection represents a necessary and important stage, as well as the different service composition types. The second part will be devoted to a review of the literature, in which we will study a panel of proposed approaches to solve the problem of QoS-aware IoT services composition, with a comparative study.

- **Chapter 4: GNN-QSC: QoS-aware IoT Services Composition Approach**

  In this chapter, we propose a new QoS-aware IoT service composition approach (GNN-QSC - Genetic Algorithms and Neural Networks for QoS-aware IoT Services Composition). This approach will be based on AI techniques, of which we will use the neuron networks as a clustering technique in order to reduce the composition space and thus reduce the composition time.

- **Chapter 5: Experimental results and performance analysis**

  This chapter is dedicated to the simulation and validation of the GNN-QSC services composition approach proposed. It is composed of two parts. The first part is devoted to an illustrative scenario to show the functionality steps of the proposed approach. However, the second is to evaluate the performance of the proposed approach through a set of experiments, in which we will pass our approach to the comparison with other existing IoT services composition approaches in the literature.

- **Conclusion and perspectives**

  In which, we present a synthesis of the main ideas of our proposals. On occasion, we take up certain reflections with the aim of highlighting the main contributions and identifying the open questions and perspectives of this work.

Chapter I

Embedded systems generalities

I.1 Introduction

Designed to operate with minimal or no human intervention. Embedded systems invade our daily lives. These systems provide us with important services, such as, when driving a car, they control its engine, brakes, and airbags. They control the take-off and landing of planes, and maintain the flight path to ensure our safety. They also control our blood pressure and the rate of our heartbeat. They are involved everywhere in our daily activities. These systems are important components and playing a vital role in many appliances, equipment instruments and home appliances. Their low cost, compressed size and simplicity in design make them very popular. This multidisciplinary sector of embedded systems is made up of a combination of several disciplines such as electronics, computer science, telecommunications networks and computer security. In this chapter we will make a non-exhaustive detour on the main axes along which research and industry are articulated in this field.We will first discuss some theoretical definitions of the subject, then consider one of the most predominant classifications in the field, to finish with an overview of the main characteristics that govern embedded systems.

I.2 Introducing Embedded systems

I.2.1 Definition of Embedded Systems

An embedded system is a combination of hardware and software to perform a specific task. That allows to the software to control the hardware, which offers the possibility of intelligent behavior and smart solutions [13]. An embedded system is therefore a dedicated computer system, designed for one or a few specific functions, often, they are part of a broader system or product, as in the case of a car's *Antilock Braking System* (ABS) [14].

It is agreed upon by almost all researchers that an embedded system can be an autonomous electronic and computer system, which is dedicated to a specific task. It does not have standard and classic inputs/outputs such as a keyboard or a computer screen. Hardware and software are intertwined, since embedded software is usually buried,

embedded in hardware. Hardware and software are not as easily discernible as in a classic PC type work environment. The term *Embedded System* can be used to designate the embedded hardware or the embedded software, or both combined.

Peter Marwedel [15] gives the following definition: *"Embedded systems are information processing systems embedded into enclosing products such as cars, telecommunication or fabrication equipment"*.

"An embedded system is a combination of hardware and software, and perhaps with mechanical add-ons or other components. Which are intended to perform a dedicated function", another definition of the embedded system given by Michæl Barr [16].

An embedded system consists of three main components [17]:

- Hardware specially designed for a specific task ;
- Application software ;
- An embedded operating system, a computer software that manages hardware and other software .

Hardware and software are always included in an embedded system. However, when it comes to the operating system, there are exceptions. Very simple embedded systems may not have an operating system [18].

Regarding the user interface, most embedded systems do not have a human-machine interface. Some may have a touch screen as a user interface. Nevertheless, there are some embedded systems that have quite complex interfaces, such as cell phones [19].

Software in embedded systems is called firmware. The programs in an embedded system are stored on chips instead of hard disks as in computers [20]. The introduction of : Arduino and Raspberry Pi (Figure I.1) has made the development of embedded systems easier.

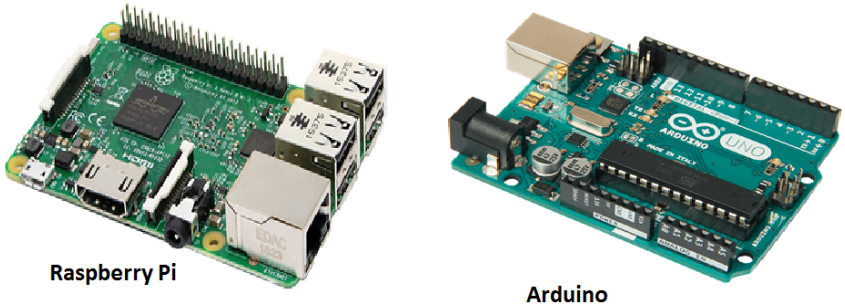

Figure I.1: Raspberry Pi 3 Model B Vs Arduino.

Today, large parts of the electronics industry are embedded systems. Most of our equipment such as washing machines, cars, planes and mobile phones have an embedded systems.

I.2.2 Some examples of embedded systems

The first system recognized as an embedded system, developed by Charles Stark Draper of MIT, is the Apollo Lunar Mission Guidance System (*Apollo Guidance Computer* (AGC)). The AGC was a real-time processing computer responsible for the inertial guidance system. The unit featured 64 kB of ROM, containing all the programs, and 4 kB of RAM. The processor consisted of more than 5000 logic gates made using integrated circuits. The whole thing weighed about 35 kg [21].

Currently, embedded systems cover all aspects of modern life, they are found in many areas and there are many examples of their use:

- **Home automation:** air conditioning, home security systems, lighting systems, etc.
- **Appliances:** microwave oven, television, dishwasher, washing machine, refrigerators, freezers etc.
- **Computer and Printing:** hard drive, CD player, photocopier, printers, copiers, fax machines, scanners, etc.
- **Transport:** Automotive, Aeronautics (avionics) etc.
- **Electronic consumable:** mp3 players, Personal Digital Assistants (PDA), digital cameras, smartwatches, digital watches, GPS navigation devices, etc.
- **Military:** radar, missile, night vision, etc.
- **Telecommunication:** telephony, mobile phone, router, firewall, gateways, etc.
- **Medical equipment:** Heart rate monitors and pacemakers.

I.2.3 Embedded Systems Constraints

Given the specificity of embedded systems, thy are subject to a set of constraints [22]:

- **The cost,** as low as possible, especially if the system is produced in large series;
- **The size,** generally the embedded systems size must be reduced;
- **The memory footprint,** or the embedded systems memory space is generally limited;
- **The lowest possible energy consumption,** due to the use of batteries as the sole power source;
- **The operational availability,** a critical system must never fail (as far as possible);
- **The security constraint,** since some information may be confidential;
- **The temporal constraint,** the execution times and the time limit of a service are known a priori. This means that such systems have real-time properties;

I.2.4 Embedded System Types

We can classify Embedded systems as follows:

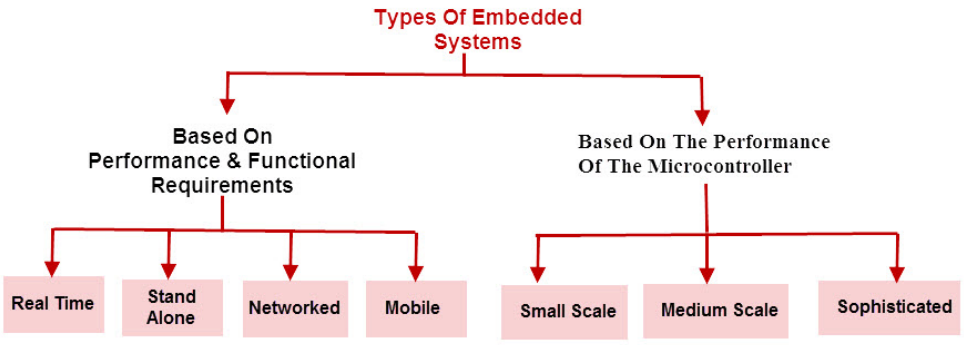

Figure I.2: Types of Embedded Systems.

According to figure I.2, embedded systems are classified in tow mains parts: based on their performance and functional requirements and based on the microcontroller performance.

I.2.4.1 According to the performance of the microcontroller

Embedded Systems are classified into three types based on the performance of the microcontroller:

Small Scale Embedded Systems

The design of these types of Embedded systems is based on a single 8 or 16-bit microcontroller that can be powered by a battery. The main programming tools for developing embedded software for small-scale Embedded systems are an editor, an assembler, a cross assembler, and an Integrated Development Environment (IDE).

Medium Scale Embedded Systems

These types of embedded systems design with a single or 16 or 32 bit microcontroller, RISCs or DSPs. These types of embedded systems present both hardware and software complexities. The main programming tools for developing embedded software for medium-scale Embedded systems are C, C++, JAVA, Visual C++, RTOS, debugger, source code engineering tool, simulator and IDE.

Sophisticated Embedded Systems

These types of Embedded systems have enormous hardware and software complexities that may require ASIPs, PLAs, scalable, or configurable processors. These types of Embedded systems are used for cutting-edge applications that require hardware and software co-design and components that need to be assembled into the final system.

I.2.4.2 According to their performance and functional requirements

Based on their performance and functional requirements, there are four categories of embedded systems [23]:

Standalone Embedded Systems

Standalone embedded systems do not require a host system like a computer, it works by itself. It takes input from analog or digital input ports and processes, calculates and converts the data and outputs the resulting data through the connected device, which controls, drives and displays the connected devices. As examples of standalone Embedded systems, we can cite: digital cameras, mp3 players, video game consoles, microwave ovens.

Networked Embedded Systems

In these types of embedded systems access to resources is related to a network. The connected network can be LAN, WAN or the internet. The connection can be any wired or wireless. This type of embedded system is the fastest growing area in embedded system applications. The embedded web server is a type of system in which all embedded devices are connected to a web server and accessed and controlled by a web browser. An example of a LAN networked embedded system is a home security system in which all sensors are connected and operate over the TCP/IP protocol.

Mobile Embedded Systems

These systems are used in portable embedded devices such as cell phones, mobiles, digital cameras, mp3 players and personal digital assistants, etc. The basic limitation of these devices is the other resources and memory limitation.

Real time Embedded Systems

A real-time embedded system is defined as a system that provides a required output within a given time. These types of embedded systems respect the deadlines for carrying out a task. Real time embedded systems are classified into two types such as soft and hard real time systems (more details see section I.2.5).

I.2.5 Real-time Embedded Systems

In a real-time system, the validity of an action depends on the time that elapses between the end of its execution and the event that triggered it. In other words, an action is valid if and only if the time that elapses between the end of its execution and the event that triggered it is less than a maximum limit.This maximum bound is called the real-time constraint (a temporal constraint) [24].

"A real-time system is a system in which the accuracy of the results depends not only on the logical accuracy of the calculations but also on the date on which the results are produced. If the temporal constraints are not satisfied, a system failure is said to have occurred" [21].

In other words, computer system that monitor, manage, or control an external environment is a real-time system. This system and its external environment are usually connected via sensors, actuators and other interfaces. As soon as they are produced, the real-time system must be able to react to these events. A real-time system is said to be clock-based when its interactions with the environment take place at times determined by the clock. However, in an event-based system, interactions are determined by the

environmental events.

Regarding the importance of temporal constraints in real-time systems, tow main categories are distinguished: *Strict or Hard real-time systems* and *Soft real-time systems* [25]. The figure I.3 illustrates a comparison between the two categories.

- **Real-time systems with Strict/Hard constraints:** strict/hard real time system does not tolerate any overrun of the temporal constraints. Such overruns can lead to critical or even catastrophic situations. Examples include flight control systems, nuclear station control systems, and railway track control systems;
- **Real-time systems with Soft constraints:** soft real time system allows certain limited overruns of temporal constraints beyond which the system becomes unusable. Multimedia applications are a good example of such systems;

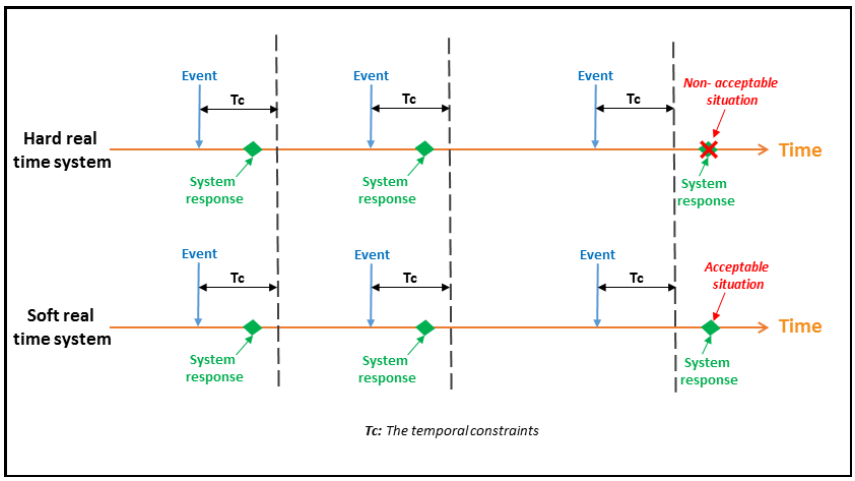

Figure I.3: Strict real-time and soft real-time comparison [26].

Real-time embedded systems are embedded systems equipped with real-time systems. In other words, the combination between embedded systems and real-time systems created other systems known as real-time embedded systems. The figure I.4 can illustrate the relationship between the two systems.

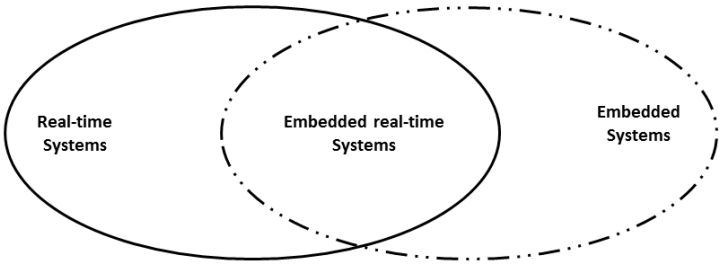

Figure I.4: Embedded real-time systems.

I.2.6 Embedded Systems Characteristics

Embedded systems encompass a number of characteristics, they can be summarized as follows [27]:

1. **Mass production:** most embedded systems are designed for a mass market. This means that the cost of a single production unit should be as low as possible;
2. **Dedicated interface:** The embedded systems interface is specially designed to meet the stated purpose and it is easy to use;
3. **Minimization of the mechanical subsystem:** for maximum reliability and minimum cost;
4. **Clarity of features:** the software implanted in the ROM must determine the functionality in detail. The quality standards for this software are high, as long as they are not changeable;
5. **Maintenance:** since partitioning the system into replaceable units is too expensive, these systems are designed to be non-maintainable;
6. **Communication and security:** connections of embedded systems with larger systems must be supported, and the topic of security is of utmost concern;
7. **Limited energy consumption:** most embedded devices are powered by batteries with a limited lifespan.

I.2.7 Architecture of Embedded Systems

Whatever function an embedded system performs, the broadest view of its architecture reveals three main components: operating system, hardware and software. Hardware that includes a central processing unit, typically in the form of a micro-controller; and a series of programs or software that manage the functionality of the hardware [28] [29]. The figure I.5 illustrates this general view by showing the main components.

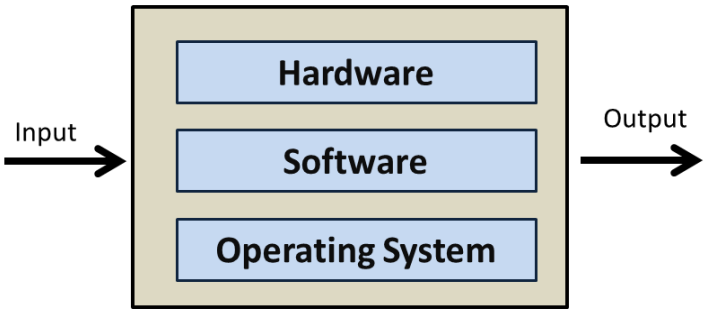

Figure I.5: General view of an embedded system.

I.2.7.1 Hardware Architecture

The electronic equipment constituting the embedded systems are placed on a circuit board. A processor, memory, data buses, and input/output devices are the main components. To perform its calculation operations, embedded systems are based on two types of processing units: microprocessors and microcontrollers. With a slightly simpler construction, the microprocessor consists of only one CPU and requires the connection of other components like memory chips. On the other hand, microcontrollers are designed as independent devices, they also have memory and external peripherals such as flash memory, RAM or a serial communication port to the processor.

Von Neumann and Harvard architectures are the two architectures for how these electronic components work together.

Von Neumann Architecture:

This type of architecture was developed by the mathematician John von Neumann in 1945. In this architecture, the design of computers consists of the residence of both program instructions and data in the same memory device. The processor and the memory device are separate from each other [30].

The von Neumann architecture consists of the use of a single memory while also having a small cache memory near the processor. It includes the following components:

1. A memory, data and instructions are both contained in memory in binary form;
2. A processing unit, mathematical and logical operations are performed using the processing unit;
3. A controller, the instructions are interpreted in memory by a controller which also ensures their execution;
4. An Input/Output peripherals, alowing the communication between the user and the control unit.

Harvard Architecture:

In this architecture different memory devices and different data buses are used for program instructions and data. The thing that will allow the processor to access both instructions and data at the same time [31].

In a computer equipped with Harvard architecture, reading instruction and access to data storage by the CPU is possible in a simultaneous manner. So using the Harvard architecture will be faster than using the von Neumann architecture, where instructions and data are stored in the same memory and are even fetched on the same data bus. The thing that will prevent the CPU from reading an instruction and performing data storage at the same time.

I.2.7.2 Software Architecture

The software is the most abstract part of the embedded system and as essential as the hardware itself. It includes programs that determine the order in which hardware

components operate.

As we saw earlier, the embedded system is usually designed to perform a single specific task, the thing that does not require important resources [32]. Therefore, the design of software for embedded systems takes into consideration:

1. A restricted memory;
2. Low processor speed;
3. Minimum power consumption.

There are several types of software architecture for embedded systems, they can be listed as follows [33]:

Simple control loop

In this architecture, the program is composed of a single loop which calls a function executing one task or another. This design can be called a Simple Control Loop or simply a Control Loop.

Interrupt-controlled system

There are some embedded systems that are mainly controlled by interrupts. This means that a specific event, for example an interrupt generated by a timer, triggers software that performs a task in the system. Usually these types of systems perform a simple task in a main loop, but this task is not very sensitive to unexpected delays. In some cases, the interrupt handler adds other tasks to a queue. Later, after the interrupt handler has finished, these tasks will be executed by the main loop. This method brings the system closer to a multitasking kernel with discrete processes.

Cooperative multitasking

A multitasking system resembles the simple control loop scheme, except that its design allows it to perform a set of tasks, and each task has its own environment for execution. If a task is inactive, it calls an inactive routine ("pause", "wait") i.e. no operation. We note that Cooperative Multitasking has the same advantages and disadvantages as for the control loop, except that we can add new software more easily.

Preemptive multitasking or multi-threading

In this type of system, depending on a timer a low-level piece of code switches between tasks or threads (connected to an interrupt). It is in this level that the system is generally considered to have an "operating system" kernel. Depending on the amount of functionality required, this introduces more or less the complexity of managing multiple tasks conceptually running in parallel.

As any code can potentially damage another task's data (except in larger systems using an MMU), programs should be carefully designed and tested, and access to shared data should be controlled by synchronization policy, such as message queues, semaphores or a non-blocking synchronization scheme. Due to these complexities, it is common for

organizations to use an *Real-Time Operating System* (RTOS), the thing that will allow application programmers to focus on device functionality rather than operating system services, at least for large systems. Due to limitations in memory size, performance, or battery life, smaller systems often cannot afford the overhead associated with a generic real-time system. However, choosing that an RTOS is required comes with its own set of issues, as the selection must be made before beginning the application development process. This forces developers to choose the embedded operating system for their device based on current requirements and thus limits future options to a great extent. Restricting future options becomes more problematic as the life of the product decreases. Additionally, the level of complexity continues to grow as devices must manage variables such as serial, USB, TCP/IP, Bluetooth, wireless LAN, network radio, multiple channels, data and voice, enhanced graphics, multiple states , multiple threads, many wait states and so on. These trends are driving the adoption of embedded middleware in addition to a real-time operating system.

Microkernels and Exokernels

A microkernel is a logical step up from a RTOS. The usual arrangement is that the operating system kernel to allocate memory and switch the processor to different threads of execution. User mode processes implement major functions such as file systems, network interfaces, etc. Generally, microkernels succeed when task switching and inter-task communication are fast and fail when they are slow.

Communication between exokernels is efficient by using normal subroutine calls. The hardware and all the software in the system are available and expandable by application programmers.

Monolithic kernels

In the Monolithic kernels, a relatively large kernel with sophisticated capabilities is suitable to fit into an embedded environment. This gives an environment similar to a desktop operating system like Linux or Microsoft Windows for programmers which makes it very productive for development. On the other hand, it requires much more hardware resources, is often more expensive, and due to the complexity of these kernels, can be less predictable and reliable. As examples of embedded monolithic kernels, we have embedded Linux and WinCE.

Exotic custom operating systems

A small fraction of embedded systems require safe, timely, reliable, and efficient behavior that cannot be achieved with any of the above architectures. In this case, the construction of a suitable system is recommended. In some cases, the system can be divided into a "mechanism controller" using special techniques, and a "display controller" with a conventional operating system. A communication system passes data between the two systems.

Additional Software components

In addition to the main operating system, many embedded systems have additional

upper-layer software components. These software layers consist of network protocol stacks such as TCP/IP, FTP, HTTP and HTTPS, and also storage capabilities like flash memory management systems. If the embedded device has audio and video capabilities, the system should contain the appropriate drivers and codecs. Many of these software layers are included in the case of monolithic kernels. Additional software components will be available in the RTOS category depending on the commercial offer.

I.2.7.3 Operating System Architecture

An operating system OS is the software that manages the hardware and other software resources of the computer and provides common services for the computer programs. Embedded systems requires operating systems. Very simple embedded systems may not have an operating system, but it's really rare for embedded systems to have no operating system. Operating systems designed specifically for embedded use are often used by embedded systems For example, for mobile phones an operating system designed specifically for their use. This system manages the user interface and all the phone basic functions [34].

An embedded operating system must be efficient and reliable. Often, to the detriment of certain functionalities, the efficiency of the embedded operating system comes. An embedded operating system has fewer features than a standard computer system. The operating system is generally adapted to the functionalities of the embedded system. Many of the usual operating system components are removed if they are deemed unnecessary or not needed.

An embedded operating system often runs on a resource-constrained hardware component such as memory. Operating systems often have a limited task tailored to run a particular program that performs a particular operation. In order to optimize the use of the CPU and take advantage of its processing power, critical code is implemented by software developers that they write into the operating system. This machine-efficient language can potentially improve speed and performance at the expense of portability and maintainability. Embedded operating systems are often written in a systems programming language such as C.

An embedded operating system can either be an operating system designed specifically for the embedded device, or one of several operating systems adapted to run on an embedded system. Common embedded operating systems are: Symbian, Windows Phone, Windows IoT and Linux.

Embedded Operating Systems Vs. Computer Operating Systems

In embedded operating systems the software is part of the operating system, so the entire software is just one executable. This is considered an important difference between most embedded operating systems and other computer operating systems. The embedded operating system can only run one program unlike PC operating systems. They are unable to load and run various applications.

Since embedded operating systems often only run a single application, the hardware is low on memory and a slow CPU is typically used. To make the most of these limited

computing resources, embedded operating systems are usually programmed in machine language. Each operating system is adapted to the hardware for which it was designed, and therefore, will not be compatible with other hardware systems with other configurations.

Commercial Operating Systems

Many operating systems exist on the market. Compared with free operating systems, they have both advantages and disadvantages.

There are many real-time operating systems on the market, many of them from well-established and reputable suppliers. However, purchasing one of these systems is something that should be carefully considered. Important factors include the size of the company, the quality of the products and its use.

Also, among the requirements is the possibility of technical support. Both buyer and seller must commit to the long term when purchasing an operating system. looking at a possible processor migration in the future is one aspect of this relationship. New versions of the system may be delivered by the provider if he is well established, or his product is probably designed to simplify upgrades. Another important point too is the good documentation and can be expected from a commercial real-time operating system vendor.

Each embedded system is technically different which is one disadvantage of commercial operating systems. From device to device CPU, memory, and external peripherals vary. In addition, the operating system must adapt to the embedded system. Commercial operating systems require licenses.

Free Operating Systems

Free real-time operating systems are often easily downloaded and very popular. Linux is not completely free because a supported version of Linux is not free. However, a supported packaged version is something most developers are likely to spend money on.

The advantage of free operating systems is that you don't have to pay anything, and you won't have to do it later, because there are no license fees. Freeware operating systems often include source code, which is useful as a reference because documentation may be limited and it may be difficult to get help later. For configuration and transfer to a new hardware environment, source code is also a requirement.

On the other hand, the disadvantage of free operating systems is that implementing it on an embedded device is a long-term commitment, so the question of long-term support is important. The thing that is not guaranteed for a free operating system, you cannot count on long-term support.

I.2.8 Some Operating Systems for Embedded systems

In what follows we will cite some operating systems designed for embedded systems.

Linux

In embedded systems, Linux can be used as the operating system. Some of the advantages of Linux as the basis for an embedded operating system include: open source,

supplier independence, low cost, and hardware support.

Windows IoT

For small secure smart devices, Windows 10 IoT Core system is designed, it supports ARM processors. Considering all the power of Windows, the advantages of developing Windows systems worldwide are shared by Windows 10 IoT.

TinyOS

TinyOS is an open-source embedded operating system and platform for low-power wireless devices developed by TinyOS Alliance. It is licensed BSD for low power wireless devices. Mostly, It is used in sensor networks, personal networks, smart buildings, and smart meters.

Contiki

The developer of Contiki is Adam Dunkels. Contiki is an operating system designed for low-power networking systems used in wireless devices in the IoT. It is a highly portable, open source system and supports multitasking for embedded systems in memory-efficient networks and wireless sensor networks. It is designed specifically to run on hardware devices that are severely limited in memory, processing power, and communication bandwidth.

Mantis

Mantis is a multi-threaded embedded wireless sensor network operating system. Mantis is a multi-threaded, open-source operating system written in C designed specifically for wireless sensor networks. The Mantis operating system provides simplified programming of wireless sensor nodes using a simple C API.

LiteOS

Developed by Huawei Technologies. Designed for wireless sensor networks. LiteOS is an open source and interactive Unix-like operating system. using the tools provided with LiteOS, it will be possible to operate one or more Unix-style wireless sensor networks, transfer data, install applications, retrieve results or configure sensors. They also allow the development of applications for the nodes and distribute these programs wirelessly to the sensor nodes.

I.3 Sensors and Actuators

I.3.1 Transducer

A transducer is a physical device that converts one form of energy into another. Mechanical transducers are transducers that convert physical quantities into mechanical quantities. We also call electrical transducers, transducers that convert physical quantities into electrical quantities [35, 36]. Transducers allow interaction between technical equipment and the physical environment using their communication interfaces and their processing units.

The following are some examples of the actions of transducers:

- Electrical energy is converted into mechanical energy or motion by an electric motor.
- Electrical signals are converted into sound waves using a loudspeaker.
- Light is produced by an incandescent lamp by converting electrical energy into optical energy.

I.3.2 Sensors

I.3.2.1 Definitions

The device that detects and responds to some type of input from the physical environment is called a sensor. Thus, the sensors act like the eyes and ears of an embedded device by detecting changes in the environment around them.

Sensors are transducers that convert physical, biological or chemical parameters into an electrical signals and can return digital data. Sensors are now an integral part of our daily lives, they are present in industrial automation and go as far as our home automation. Common sensors are temperature sensors, humidity sensors and gas sensors [37]. A sensor is a device (usually electronic) that has the ability to detect events or changes in its physical environment (for example, temperature, sound, heat, pressure, flow, magnetism, motion and chemical and biochemical parameters) and provide corresponding output [38]. Most sensors take analog data as inputs, after processing, they deliver digital outputs, often electrical. Since the sensing element ,on its own, gives an analog output, an analog-to-digital converter is often required. The figure I.6 gives an overview of a sensor components.

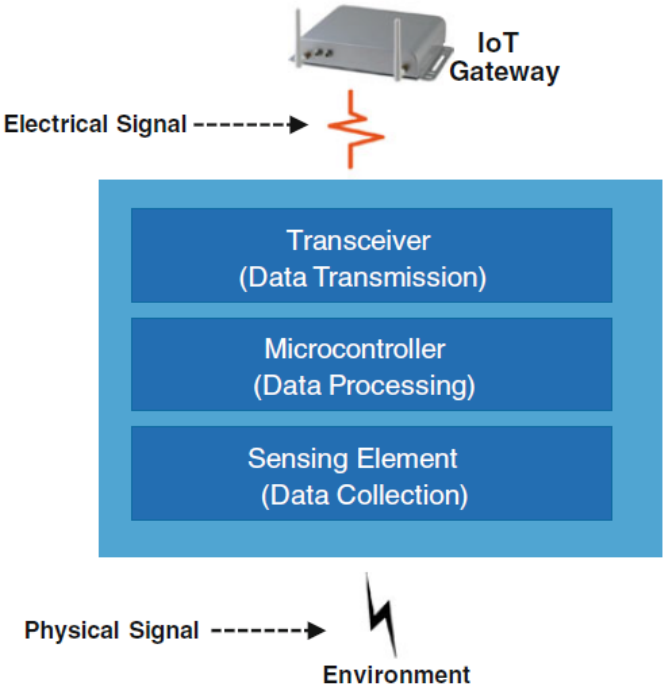

Figure I.6: Sensor components overview.

Sensors are considered key components of IoT. Their importance lies in taking measurements, collecting and transferring data, and in the work of monitoring processes. The role of sensors is to detect events or changes in the environment and send this information to another location, for example, a server or a web page. They are designed to measure physical quantities, then transform them into numerical values readable by humans or usable by data systems.

Sensor technology is developing ever faster due to new discoveries in materials and nanotechnology. The result continues to be felt, increased accuracy, reduced size and cost, and the ability to measure or find values that had not been possible before. In fact, sensor technology is developing so rapidly and becoming more and more advanced that we will see billions of new sensors in production every year within a few years. The low price of most sensors helps reduce IoT expenses and enables the use of large-scale embedded systems. The IoT has upset the manufacturing industry, and sensors are at the heart of using the IoT.

I.3.2.2 Sensors Use

At present, sensors are widely used and there are hundreds of different types of sensors. There are temperature sensors, flow sensors, voltage sensors, humidity sensors, etc.

For example, autonomous vehicles are full of sensor technologies and have several sensors to measure various quantities such as power, load, torque, motion, speed, displacement, position, vibrations and shocks [39].

Using sensors in a factory's production equipment can help identify bottlenecks in a manufacturing process. By identifying and addressing these bottlenecks, factories can reduce lost production time [40]. Instead of standard preventive machine maintenance, predictable maintenance by relying on sensor data to predict fairly accurately when machines need maintenance [41].

I.3.2.3 Sensors Data Storage

The decision to store sensor data locally or in the cloud is often a dilemma. There are pros and cons of both options [42].

Fast access to our data is one of the main advantages of local storage. Storing data on locally hosted hard drives is faster than uploading it to the cloud. Total control of our backups is another strong point of local storage, which allows us better control of who accesses our data. Likewise, isolating disks from the network helps protect our data against attacks.

Except that, backing up sensor data to the cloud has also many benefits. On the one hand, its cost effective and maintenance is not an issue as cloud services providers handle all upgrades and fix any issues that may arise. On the other hand, cloud storage is scalable. When you need to increase storage space, it's as simple as letting your provider know. You can decrease or increase your storage space as needed.

In the event of a disaster on site, the data will remain safe. Securing data in the cloud means you don't have to worry about losing backups of your data.

Accessibility is also a benefit of cloud storage. Data stored in the cloud is easily accessible on any device with an internet connection. by connecting to your cloud account and the data is there when you need it.

Security and privacy are the main disadvantages of cloud storage, which always raises concerns about storing valuable and important data remotely. When adopting cloud technology, it is important to be aware that you are providing sensitive information to a third party, the cloud service provider, which could potentially pose a risk. Another disadvantage of cloud storage is the lifetime cost. By storing our data in the public cloud, the costs of the services increase over time and tend to accumulate which makes additional charges. Likewise, If your applications are on-premises and your data is in the cloud, this can increase networking costs.

I.3.2.4 Types of Sensors

There are many different types of sensors in various technologies. The most common of which include the following [38]:

Temperature Sensors

Temperature sensor is the widely used type of sensor. It measures the temperature or heat of a given medium. On the market, there are several types of temperature sensors (Thermocouple, Resistance Temperature Detector and Thermistors). The use of temperature sensors is in almost all areas, ranging from simple thermostats to highly sensitive semiconductors that are capable of controlling complex processes [43].

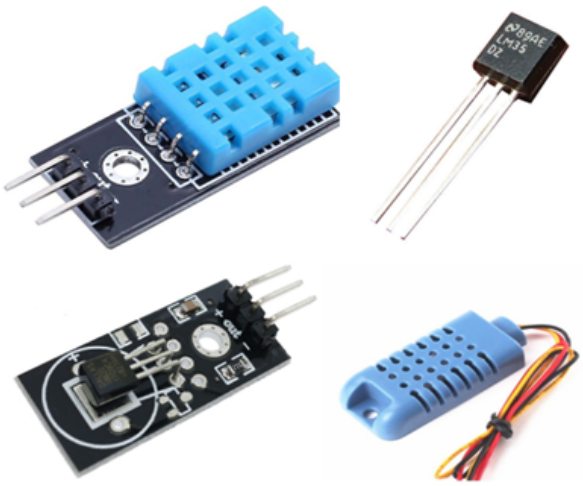

Figure I.7: Examples of temperature sensors.

Pressure Sensors

A pressure sensor is a device that detects pressure and converts it into an electrical signal. It can be used to measure the pressure of a gas or a liquid by converting the pressure as a physical quantity into an electrical signal. Pressure sensors are also useful for measuring other variables such as speed and height.

The most popular pressure sensors used in IoT systems are barometers and pressure gauges. To make weather forecasts, barometers are used, they are able to accurately

measure the air nearby. While pressure gauges are used in industrial buildings, where they are used to monitor pressure in closed environments. They can even be used in very different fields. For example, in touch screens and biological medical instrumentation, or in industry such as the automotive industry [44].

Figure I.8: Examples of pressure sensors.

Proximity Sensors

A proximity sensor is used to determine the distance to a nearby object without having physical contact with it. For this, electromagnetic radiation or radar is used that allows it to detect movements or obstacles. Proximity sensors are good for motion detection. Proximity sensors are often a primary component of equipment that involves safety, security or efficiency. They are therefore empolyed in vehicles to detect obstacles ahead when in motion. They can also be found in parking systems, museums, airports, etc. For example, when parking, parking sensors are proximity sensors designed for vehicles to alert the driver to obstacles. A Ground Proximity Warning System (GPWS) is a system responsible for alerting pilots if the aircraft is in immediate danger of flying into the ground or an obstacle [45].

Figure I.9: Example of Proximity sensor.

Motion Detector Sensors

We use the motion detector, which is an electronic device capable of detecting physical movement in an area and transforming it, whether it is the movement of an object or the movement of people, into an electrical signal. Therefore, they play a very important role in the security industry. For example, they are used in areas where there should never be any movement at all. By installing these sensors, it will then be possible to detect if someone or something is moving. They are used in several fields, such as intrusion detection, door

control, automatic parking systems, automated washbasins, toilet fans, hand dryers and automated lighting [46].

Figure I.10: Example of Motion Detector Sensors.

Optical Sensors

To detect electromagnetic energy such as light, electricity and similar elementary particles fiber optic sensor technology is used. Optical sensors can thus send, receive and convert light energy into electrical signals. Fiber optic sensors can be found in energy, healthcare, aviation, chemical and environmental IoT platforms. Optical sensors can be ideal for environments such as petroleum refining, mining operations, pharmaceutical production and chemical processing [47].

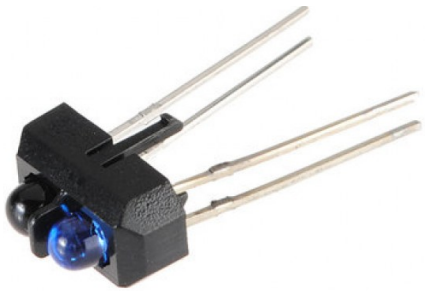

Figure I.11: Example of Optical Sensor.

Imaging sensors

Imaging sensors are used to measure image information by capturing and then converting the varying attenuation of waves into electrical signals. They are sophisticated sensors, typically used in digital cameras, medical imaging devices and night vision equipment [48].

Infrared Sensors

An infrared sensor is an electronic sensor that measures infrared light emitted by a nearby object. Infrared sensors are used in many IoT projects, especially in healthcare systems, as they help monitor blood flow and blood pressure monitoring. they can even be found in several common smart devices, such as smart watches and smartphones. Other common applications include household appliances and remote control, breathing analysis, IR visualization (i.e., visualizing heat leakage in electronics, monitoring blood flow,

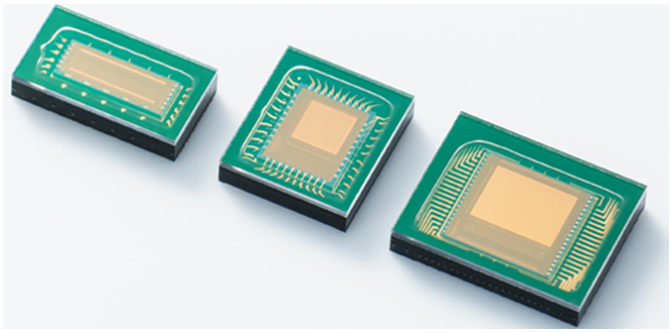

Figure I.12: Examples of image Sensor.

looking below the surface of paintings), optical communication, non-contact temperature measurements, and blind angle detection [49].

Figure I.13: Example of Infrared Sensor.

Moisture and Humidity Sensors

Moisture and humidity sensors (sometimes called hygrometer sensors) are used to detect and measure the relative humidity in the air. They use capacitive measurement based on electrical capacitance [50].

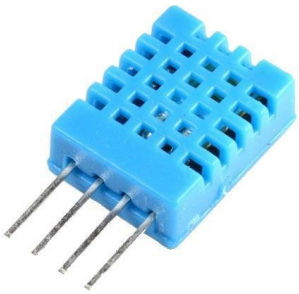

Figure I.14: Example of Humidity Sensor.

Noise Sensors

Adverse effects on humans as well as animals (eg hearing loss) can be caused by loud noise, especially in enclosed spaces. Machinery, planes, trains, construction and loud music are examples of sources of this noise. In such a context, noise sensors constantly monitor the noise levels in the surrounding environments. They send an electrical signal to a global ambient noise system to take action when they detect a change in noise level. Such an action can be an automatic one (for example, adjusting the music level) or either a simple notification [51].

Figure I.15: Example of Noise Sensor.

There are so many other types of sensors that we cannot list them all. Examples include acceleration sensors, biosensors, gas and chemical sensors, mass sensors, tilt sensors and force sensors, etc.

I.3.3 Actuators

I.3.3.1 Definitions

Actuators are transducers that operate in the opposite direction to the sensors. It takes electrical signals and converts them into physical action. As an example of an actuator, the electric motor that creates motion. Simply put, actuators are devices that accept a control command and produce a change in the physical system by generating force, motion, heat, flow, etc. We can say that the actuators are considered as the hands of the embedded systems.

An actuator is the functional part that connects the information processing of an electronic control system to a technical or non-technical system, e.g. biological process. To be able to control the flow of energy, mass or volume, actuators can be used whose output quantity is energy or power, often in the form of a mechanical working potential "force times displacement" [52].

An actuator is a type of motor responsible for controlling or acting in a system. It takes a source of data or energy as inputs (e.g. hydraulic fluid pressure, other energy sources) and converts them into motion (outputs) to control a system [38, 53].

An actuator is a mechanism responsible for controlling or moving something. To be able to do this, an actuator requires a control signal and a power source. The energy source can take several forms such as electric current, hydraulic fluid pressure or pneumatic pressure. It responds by converting energy into mechanical motion once the command signal is received.

I.3.3.2 Use of Actuators

By using an actuator, you have the ability to perform many different tasks, including robot control, household activities such as watering flowers, camera control, unmanned aircraft, and more. Among the tasks for which electric motors are often used is the generation of rotation around a fixed axis to drive the wheels, pumps, belts and robot arms.

Three types of commonly used motors can be distinguished, namely servomotors, DC motors and stepper motors.

I.3.3.3 Types of Actuators

Electric Actuators

Electric actuators are devices driven by small motors that convert electrical energy into mechanical torque to control certain equipment requiring multi-turn valves or to control valves. Electric actuators can also be found in engines to control its various valves [54].

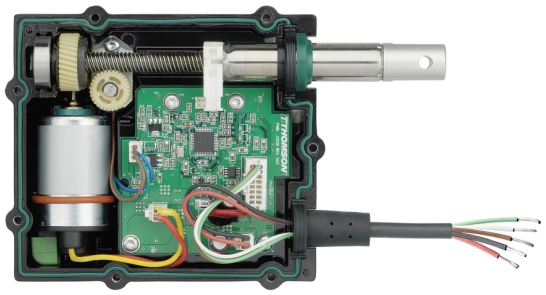

Figure I.16: Example of Electric Actuator.

Hydraulic Actuators

The design principle of hydraulic actuators is based on Pascal's law: when there is an increase in pressure at any point of a confined in-compressible fluid, then there is an equal increase at any point of the container. Hydraulic Actuators are simple devices with mechanical parts that are used on linear or quarter-turn valves comprising a cylinder or fluid motor that uses hydraulic energy to enable a mechanical process for generating an output in terms of linear, rotary or oscillatory motion. Hydraulic actuators can be operated manually, like a hydraulic car jack, or they can be operated by a hydraulic pump, which can be seen in construction equipment such as cranes or excavators [55].

Figure I.17: Example of Hydraulic Actuator.

Pneumatic Actuators

In this type of actuators (pneumatic) compressed gas is used instead of liquid in hydraulic actuators. Both types of actuators work on the same concept [55, 56].

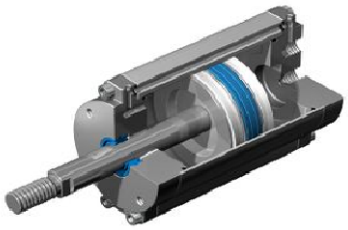

Figure I.18: Example of Pneumatic Actuator.

Mechanical Actuators

Based on the combination of mechanical structural components, such as gears and rails, or pulleys and chains, the working principle of a mechanical actuator is to convert one type of motion, such as rotational motion, into another type of motion, such as linear motion [57].

Figure I.19: Example of Mechanical Actuator.

I.4 RFID technology

Another technology to capture information from "Things" is to use *Radio Frequency Identification* (RFID), which is not a sensor but rather a mechanism. RFID makes it possible to capture information pre-embedded in the so-called tag of a thing or object using radio waves.

I.4.1 RFID System

RFID systems are composed of three main components: RFID tags, reader, application system [58, 59], as shown in Figure I.20.

I.4.1.1 RFID tags

We can also call it transponder (transmitter), it is attached to the object to be identified or counted. There are two types of tags, active or passive. Tags that are partially or fully battery powered are active tags, they have the ability to communicate with other tags and

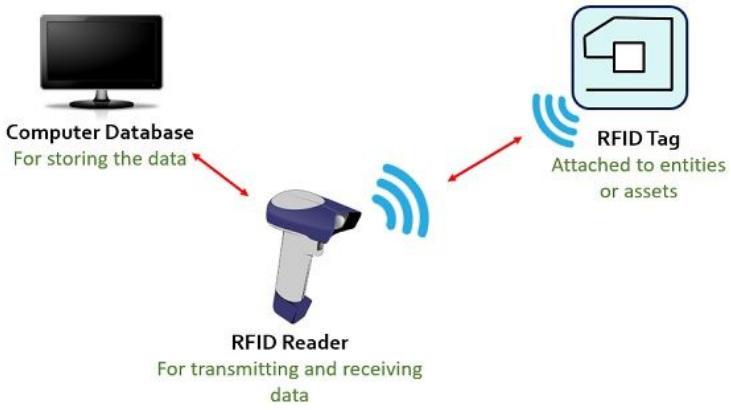

Figure I.20: Components of a RFID system.

can initiate a dialog with the tag reader. On the other hand, passive tags do not need an internal power source but are powered by the tag reader itself. Whose main purpose is to store data, the tags mainly consist of a coiled antenna and a microchip.

I.4.1.2 RFID Reader

Also known as a transceiver (transmitter/receiver) consisting of a control unit and a radio frequency interface module (RFI). Its main functions are: to activate the tags, to structure the communication sequence with the tag, and to transfer data between the application software and the tags.

I.4.1.3 Application system

Application system or the data processing system. It can be an application or a database, depending on the need. The application software is responsible for initiating all readers and tags activities. RFID provides us with a reliable, fast and flexible way to electronically detect, track and control items. RFID systems use radio transmissions to send energy to an RFID tag while the tag transmits a unique identification code to a data collection reader linked to an information management system. The data collected from the tag can then be sent either directly to a host computer or stored in a portable drive and later downloaded to the host computer.

I.4.2 RFID Technology Applications and Benefits

There are three functions of RFID system, they generally include the following aspects: monitoring, tracking and supervision. Monitoring means being aware of the system state, repeatedly observing particular conditions, especially to detect them and give warning of changes. Observation of moving people or objects is the tracking, providing a timely ordered sequence of respective location data to a model. Whereas, supervision is the monitoring of changing behaviors, activities or other information, usually of people. This is sometimes done in a secret or inconspicuous way. [60, 61, 62].

RFID applications in daily life are many and far-reaching, the most interesting and successful ones include those for supply chain management, production process control and object tracking management. Today, RFID has been gradually and widely used in

the following fields.

- Logistics and Supply;
- Manufacturing;
- Agriculture Management;
- Health Care and Medicine;
- Military and Defense;
- Environment Monitor and Disaster Warning;
- Transportation and Retailing;

For example, in automobile manufacturing, RFID technology offers a number of applications in this industry. An RFID-based vehicle anti-theft device is a protective device installed in many cars. In automotive assembly and manufacturing processes, RFID technology also holds great promise, especially for flexible and agile production planning, spare parts and inventory management.

In addition to the automation of the entire assembly process in which a significant reduction in costs and losses can be achieved, RFID technology also offers improved services to automotive users, including the automated generation of maintenance reminders and more efficient spare parts ordering. The advantages that RFID offers to the automotive industry, both for the production process and for end users, are visibility, traceability, flexibility and increased security.

I.5 Conclusion

In this chapter, we have introduced general information on embedded systems by reviewing some concepts and technologies governing this field. Then, we focused on Sensors and Actuators as an important part of embedded systems by giving their definitions, types and applications. After that, we did a passage on RFID technology,its use and its benefits.

Embedded systems, essentially the sensors and actuators networks and RFID systems, will be the main core of a recent and rapidly growing technology trend. These technologies, by combining with the Internet, form an extended network in order to computerize and intelligentize entities or objects, what we call the *Internet of Things* (IoT). The IoT which is considered as the global network composed of many connected devices or objects, will be the subject of our next chapter.

Chapter II

IoT: Theoretical bases, principles and technology

II.1 Introduction

As we saw in the previous chapter, embedded systems are an integral part of every modern electronic component. These are low power consumption units that are used to run specific tasks, sensors, actuators, RFID tags, etc. are used in various daily life applications.

Embedded systems will also be the basis for the deployment of the *Internet of Things* (IoT). When embedded systems are connected to the Internet on a large scale, it will lead to what we call the Internet of Things. The IoT means that all embedded devices that are connected to the Internet can communicate with each other without human involvement. The IoT is a relatively new concept. It is a growing field, and its development requires the support of some innovative technologies. In this chapter, we present the concepts, the theoretical bases, the principles, the characteristics and the different technologies involved in the IoT.

II.2 Internet evolution to the IoT

II.2.1 Origin and Definition

The term "Internet Of Things" was used by Kevin Ahston, the co-founder of the Auto-ID Center at MIT, in 1999, as part of a presentation for the company Procter & Gamble (P&G). This term summons the world of objects, devices and sensors that are interconnected by the internet [63].

IoT was originally proposed to refer to objects connected and uniquely identified with Radio Frequency Identification (RFID) technology [64]. The RFID technology, considered one of the main tools of the IoT, allows microchips to transmit credentials to a reader through wireless communication. In addition, *Wireless Sensor Networks* (WSN), which mainly uses interconnected smart sensors to sense and monitor the environment, is another core IoT technology [65]. The tremendous advances in information and communications

technology, as well as the embedded systems (RFID, WSN, etc) are contributing significantly to the development the novel concept of IoT [66]. In addition, many other technologies are used to build an extensive IoT support network [67]. Figure II.1 illustrates the evolution of IoT technologies.

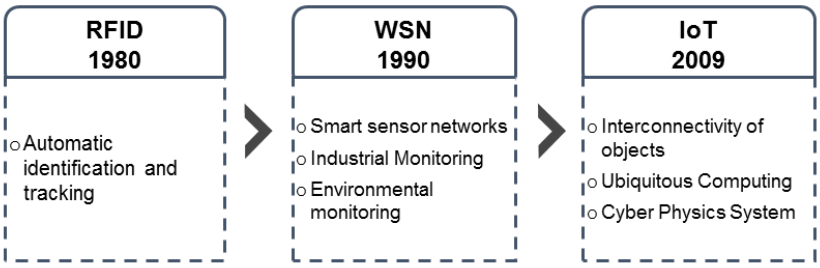

Figure II.1: IoT Technologies Evolution [64].

We can define *Internet of Things* (IoT) as a network of connected devices where each device has an IP address and the built-in technology allows it to communicate with other devices on Internet. IoT is a scenario in which objects, people or animals are equipped with unique IP addresses so that they can transmit data over the network without everyone participating.

The authors in [59] give the following definition: *"Internet of Things (IoT) is a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. It will offer specific object identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability".*

The IoT is a vast network of connected objects, animals and people. They come in all shapes and sizes, from smart microwave ovens that automatically cook at the right time, to self-driving cars, to wearable fitness devices that can track your heart rate and steps taken throughout the day. Then, this information is used to recommend a suitable training plan for you. The IoT bridges the gap between the physical and digital worlds, helping to improve the quality and productivity of people, communities and industries [68].

Based on the complexities of an IoT system, the IEEE IoT Initiative provides two definitions [69]. For low complexity systems, IEEE IoT Initiative gives the following definition :

"An IoT is a network that connects uniquely identifiable "things" to the Internet. The "things" have sensing/actuation and potential programmability capabilities. Through the exploitation of unique identification and sensing, information about the "thing" can be collected and the state of the 'thing' can be changed from anywhere, anytime, by anything."

For very complex systems, the IEEE IoT Initiative definition is the following:

"Internet of Things envisions a self-configuring, adaptive, complex network that interconnects 'things' to the Internet through the use of standard communication protocols.

The interconnected things have physical or virtual representation in the digital world, sensing/actuation capability, a programmability feature and are uniquely identifiable. The representation contains information including the thing's identity, status, location or any other business, social or privately relevant information. The things offer services, with or without human intervention, through the exploitation of unique identification, data capture and communication, and actuation capability. The service is exploited through the use of intelligent interfaces and is made available anywhere, anytime, and for anything taking security into consideration"

Both definitions have implied that IoT has four fundamentals: sensing, actuation, computing, and communication [37].

II.2.2 IoT Environment dimensions

The Internet is gradually transforming into a network formed by multitudes of connections, connecting millions of human beings, but also billions of objects [70, 71, 72]. It becomes the most powerful tool ever invented by man to create, modify and share information. This transformation shows the evolution of the Internet network from a network of computers to a network of personal computers, then to a nomadic network integrating communication technologies. The IoT gives communication technologies a new dimension, in the sense that, thanks to it, communication is not only possible anywhere and anytime, as was already the case, but also with any what object [63]. Figure II.2 illustrates the three dimensions of the IoT.

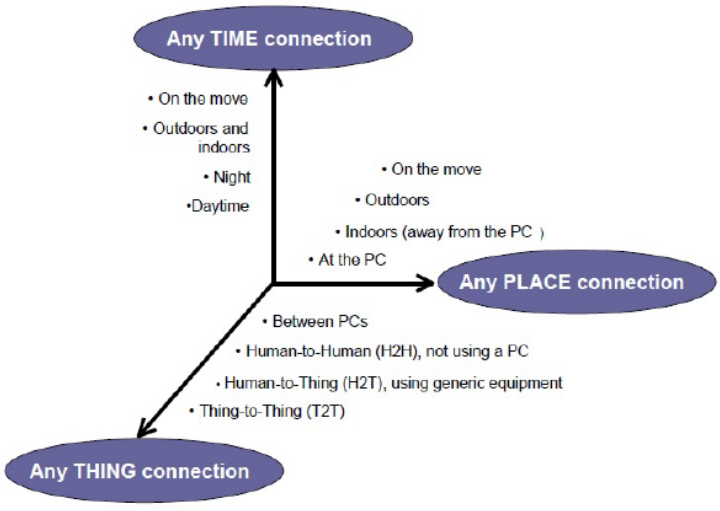

Figure II.2: Dimensions of the IoT [63].

II.2.3 Model for Internet of Things

A complete IoT model consists of five key components:

- Sensors or units with sensors.
- Actuators.

- Connection to a network.
- Data management.
- A user interface.

Sensors or Units with Sensors

The IoT systems are often used for processing measured data. Collecting data from internet-connected IoT devices around the world forms the basis for control and optimization. IoT system data is collected from the environment where the sensors are placed. It can be something as simple as taking a temperature reading or something more complex like recording a video [73].

Connection

Sensor data is sent to the cloud, but it needs a way to make it happen. Sensors and devices can connect to the cloud using a variety of methods, including Wi-Fi, satellite, Low-Power Wide-Area Network (LPWAN), and mobile, or by connecting directly to the Internet via Ethernet. Each connection option has its pros and cons in terms of power consumption, rang, and bandwidth. Choosing the best connectivity option depends on your IoT application, but all options attempt to do the same job of getting data to the cloud [74].

Data Management

Once the data reaches the cloud, the software performs some form of data processing. It can be very easy, for example, checking if a temperature reading is within tolerance range, and as it can be more complicated, like using video software to identify objects like an intruder in a building. The user must decide what to do if the temperature is too high or if an intruder breaks into the building. Therefore, the system must send information to the user [75, 76].

User Interface (Mobile Apps)

A mobile application(also known as mobile apps) is a software program developed for mobile devices such as smartphones and tablets. The IoT is a network of Internet-connected devices, each with an IP address, which communicates with users through mobile applications on smartphone interfaces. Information from IoT devices must be presented to end users. This can be done through notifications such as emails or SMS. Additionally, users can have an interface that allows them to explore the system. For example, users can view video footage around the house through a phone app or web browser. In many cases, the user can interact with the system. For example, users can remotely adjust the refrigerator temperature via an app on their mobile phone. Furthermore, some tasks may be performed automatically [77, 78].

To sum up, an IoT system consists of sensors and actuators that communicate with the cloud via some forms of connection. Once the data arrives in the cloud, the software processes it and can then decide to perform an action, such as sending an alert to the user or automatically adjusting the system without notifying the user. However, if user

access is required or the user wishes to verify the system, a user interface can be used. Any customization or action performed by the user is then sent in the opposite direction through the system. A message from the user interface is sent to the cloud and then sent back to the sensor or actuator to perform some sort of modification.

II.2.4 Generic IoT systems architecture

Designing an architecture for IoT systems mainly aims to implement a unified network of smart objects and users (if necessary), which are able to communicate with each other in a universal and ubiquitous way. For this, several researches have proposed reference architectures for the IoT [79, 80, 81, 82, 83].

Communities do not adopt a formally identical architecture, however, all proposals involve four components: smart objects, connectivity, data processing and application. Some use cases may involve additional layers, but these four components represent the foundation for all proposals. Figure II.3 illustrates the architecture of an IoT system.

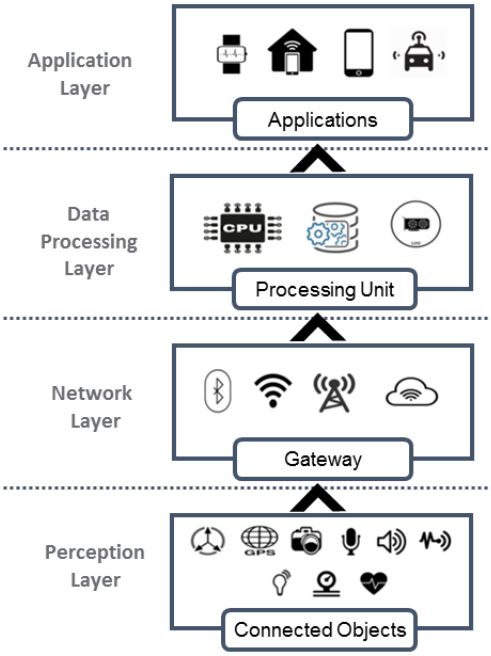

Figure II.3: Architecture of IoT systems [83].

II.2.4.1 Perception Layer

Also called the detection layer, the main objective of this layer is to identify any phenomenon within the perimeter of the sensors and to obtain data from the real world. It encompasses all the smart objects that collect data from the environment and send it to the network layer.

II.2.4.2 Network Layer

The network layer acts as a communication channel to transfer the data, collected in the perception layer, to the higher layers. This layer is implemented using various communication technologies (Wi-Fi, Bluetooth, Zigbee, Z-Wave, LoRa, cellular network, etc.).

II.2.4.3 Data Processing Layer

Processing consists of two main processes: Analyze and Store. This layer takes the data collected in the perception layer and analyzes it to make decisions, in some cases the data processing layer also saves the result of the previous analysis to improve the user experience. This layer can share the data processing result with other devices connected through the network layer [84].

II.2.4.4 Application Layer

The application layer implements and presents the results of the data processing layer to realize various applications and provide various intelligent services to users. There are various applications of IoT, such as smart transportation, smart home, healthcare, etc.

II.2.5 Internet of Things System Characteristics

The IoT is a global infrastructure for information, offering advanced services by interconnecting physical and virtual elements based on existing information and communication technologies. The IoT represents a convergence of several fields and technologies. Nevertheless, an IoT system imposes specific constraints in terms of connectivity, computing power and energy consumption, which makes it significantly different from all existing systems. The IoT is a complex system that has a number of characteristics. Its characteristics vary from one domain to another [85, 86].

Interconnectivity

The IoT systems allow all objects to connect to a global information and communication infrastructure. The Interconnectivity of these objects is essential to the provision of advanced services. These object-level interactions contribute to the collective intelligence of IoT networks. The Interconnectivity provides network access and then ensures compatibility.

Things-related services

The IoT is capable of providing object-related services within object boundaries, such as privacy protection and semantic consistency between physical objects and their associated virtual objects. In order to provide object-related services within object boundaries, the technologies of the physical world and the information world will change.

Devices heterogeneity

The heterogeneity of the IoT is one of its main characteristics. IoT devices are heterogeneous as they are based on different hardware, software and network platforms. The IoT architecture must support connectivity between heterogeneous networks. Modularity, extensibility and interoperability are central requirements for the design of heterogeneous objects in the IoT.

Large Scale

The number of objects used in the IoT network can vary from a few entities to several tens of thousands. These objects must be able to self-organize on a large scale and be

effective regardless of their number. For this, IoT network management protocols must be able to operate and adapt according to the number of objects. Also, the management of data generated from these devices and their interpretation for application purposes becomes more critical.

Dynamic Changes

Grace to the dynamic changes that occur around connected objects, the state of the devices changes dynamically, e.g., sleeping and waking up, connected and/or disconnected, as well as the context in which these devices operate (location, speed, etc.). Moreover, the number of devices can change dynamically.

Energy Optimization

Most IoT devices are battery equipped with limited energy. Often, these things are deployed in areas that are difficult for humans to access. It is therefore difficult to be able to change these batteries. Therefore, optimizing energy consumption is a main constraint.

safety

IoT objects are connected to the Internet, therefore there are significant security risks, especially with regard to the confidentiality and authenticity of data. One of the best examples of security requirements is the need to maintain user privacy against any information collection. As creators and recipients of the IoT. We must not forget about security. We must design the security of our personal data and the security of our physical well-being (endpoints, networks, data, etc.).

II.2.6 IoT Application Areas

Thanks to the rapid advancements in the underlying technologies, the IoT opens up enormous possibilities for a large number of applications that promise to improve the quality of our lives. Several criteria have allowed the expansion of the use of the IoT [87]:

— The reduction in the sensors cost;

— The connectivity explosion;

— The increase in the computing power of processors;

— The miniaturization of its elements;

— Cloud development.

It is difficult to consider all IoT applications taking into account the evolution of technology and the various users needs. IoT applications can be found in several important areas. In this section, we present the most important IoT applications [88].

IoT Smart Manufacturing/Industrial

The benefits of IoT are great in the industrial world, and *Industrial Internet of Things* (IIoT) was born. This new technology allows for better production quality and full monitoring of the entire production chain, resulting in increased productivity and efficiency, as well as other economic benefits [89, 90].

IoT Smart Transport/logistics

Means of transport are increasingly equipped with smart sensors/actuators. The roads themselves and the merchandise transported are also equipped with tags and sensors that send important information to traffic control sites and transport vehicles in order to make decisions (better direct traffic, help with depot management, monitor the status of goods transported, etc.) [91, 92, 93].

IoT Smart Energy

The IoT is revolutionizing nearly every sector of the energy industry, from power generation to transmission and distribution, transforming the way energy companies interact with their customers. With today's growing demand for process automation and operational efficiencies, it's hard to underestimate the current impact of the Internet of Things on the energy industry. Companies are increasingly addressing IoT use cases in energy management, such as optimizing consumption with smart meters, monitoring and maintaining energy systems, improving efficiency of energy systems, improvement of operational safety and prevention of production accidents [94, 95, 96, 97].

IoT Smart Retail

More and more retailers are realizing that they can improve their profitability and in-store customer experience through innovative IoT use cases. More and more retailers are digitizing their stores and creating smarter processes. Common IoT solutions for retail include in-store digital signage, customer tracking and retention, merchandise management, inventory management, smart vending machines, and more [98, 99, 100, 101].

IoT Smart Cities

Entire cities are becoming digitally connected and smarter through the power of the Internet of Things. Cities improve the lives of their citizens by collecting and analyzing large amounts of data from IoT devices in various city systems. Smart cities can make better decisions thanks to data collected on infrastructure needs, transport needs, crime and security. Studies show that cities improve quality of life indicators (crime, traffic, pollution, etc.) by 10-30 % using existing smart city applications [102, 103, 104].

IoT Smart Healthcare

The Internet of Things has been slow to become ubiquitous in healthcare. However, given the epicenter of the COVID-19 pandemic, things seem to be changing. Early data suggests digital health solutions related to COVID-19 are on the rise. Some IoT healthcare applications such as telemedicine consultation, digital diagnostics, remote monitoring and robotic assistance are in high demand. IoT healthcare applications can help: Reduce emergency wait times; Track patients, staff and inventory; Strengthen medication management; Ensure the availability of critical equipment. The IoT has also introduced several wearable devices and equipment that greatly facilitate the lives of both patients and doctors [105, 106, 107].

IoT Smart Supply Chain

IoT devices have revolutionized supply chain management. As supply chains extend

ever further to the end customer, more complex flows of goods emerge and delivery becomes more complex. Logistics service providers are increasingly integrating networked digital solutions to address this complexity. In supply chains, Internet of Things devices are an effective way to track and authenticate products and shipments using GPS and other technologies. Additionally, the storage status of products can be monitored, improving quality control throughout the supply chain [108, 109, 110].

IoT Smart Agriculture

A smart greenhouse is one of many ways to solve this problem. Smart greenhouses are a revolution in agriculture, creating a self-regulating micro-climate for plant growth using sensors, motors, monitoring and control systems that improve growing conditions and automate the growing process [108, 111, 112].

IoT Smart Homes/Buildings

Smart buildings have an incredible ability to connect people and technology. Smart building technologies not only support building management, but also provide valuable insight into how building spaces are used and enjoyed. This contributes to energy efficiency, building sustainability and human resource management efforts. Smart home, also called home automation, is the trend in this sector [113, 114, 115].

II.2.7 Internet of Things challenges

The Internet of Things continues to grow. As time passes, the number of objects increases, which will lead to complex technological challenges concerning security, interoperability, scalability, transparency, self-adaptation, and energy management [116, 117].

- **Security:** Security is very important in the Internet of Things as it will become more and more ingrained in our lives. Concerns will no longer be limited to information protection, our life and health may become the target of attack. In addition, the interconnection of objects generates problems of confidentiality, integrity and authenticity [118, 119].
- **Scalability (scaling up):** The number of connected objects in the IoT is constantly growing, exhausting the number of addresses available for possible use of the IPv4 protocol for addressing. The IPv6 protocol is a solution to this problem, by offering 2^{128} different addresses [120]. This allows a very high level of connectivity. On the other hand, transmitting, storing and processing the data of all these objects is a major challenge [121].
- **Energy:** A connected object is limited in energy because of its size. In addition, a connected object must save its energy consumption to avoid shortage problems and reduce the energy cost that can be generated by the expected number of connected objects. The energy consumed by an object is mainly due to the following operations: data acquisition, data processing and communication. All of these features pose challenges for developers to design applications and protocols to optimize power consumption [122].

- **Usability:** Miniaturized objects are not equipped with input interfaces (such as keyboards and mice) or displays (such as screens) which represents a major challenge for researchers who must design applications allowing the user simple to interact with the different objects without too much difficulty [123, 124].
- **Self-adaptation:** Self-adaptive mechanisms are necessary in order to ensure the continuity of services and meet the expectations of users according to their contexts and the dynamics of the environment. The integration of self-adaptation mechanisms in a solution is the key to managing changes in the context and makes it possible to increase the degree of autonomy, robustness and flexibility of such a solution [125, 126].
- **Heterogeneity and Interoperability:** Participating objects in an IoT network are heterogeneous in several aspects, namely: hardware components, software components, technologies and communications protocols. To address this last challenge, a movement from the Internet of Things to the *Web of Things* (WoT) was born [127, 128].

II.3 IoT communication technologies

The Internet is made up of a variety of technologies that communicate in real time with a wide variety of devices, from single devices to large embedded technology platforms and connected cloud systems. Connecting components over the Internet requires a protocol that allows devices and servers to communicate with each other. Many connections are needed to connect widely fragmented IoT networks. Both wireless and wired technologies are used in IoT communication. Also use Ethernet if the device is not particularly mobile [129].

Companies want to take advantage of IoT, which will lead to more IoT applications in the future. With an ever-growing number of providers looking to connect homes, factories, vehicles and medical services in smarter ways, it's important to understand the different standards that apply. An IoT solution in its simplest form is a set of sensors and actuators connected to a central management unit that allows a user to perform an action in the environment. IoT devices must use the same standard protocol in order to communicate with each other and transfer data. Different standards make communication between devices difficult. Standards compatibility is therefore important for creating reliable and large-scale IoT networks. There are now multiple IoT communication protocols and standards to simplify IoT design and increase vendors' ability to innovate quickly [130].

II.3.1 Wireless Connectivity Technologies

Wireless communication technology can be defined as a technology that enables wireless transmission of data or information from one device to another. IoT systems can transmit information from one device to another over transmission channels with distances ranging from a few centimeters to hundreds of kilometers, depending on the area of application.

The IoT is a network of objects that can communicate with each other. The most important aspect of this is the connection between them. How are these devices connected and how are they supposed to communicate?

Today, thanks to the diversity of the market, there are several wireless transmission technologies as well as those available. Below we list the four basic signal types used for wireless transmission in IoT systems. These are chosen based on their popularity in the available literature and publications: Radio Frequency Transmission, Infrared Transmission, Microwave Transmission and Lightwave Transmission [131]

II.3.1.1 Bluetooth

Bluetooth, defined in the category Wireless Personal Area Network (WPAN), was developed by the Bluetooth Special Interest Group (Bluetooth SIG) and is a type of wireless connection intended for short-range communication between devices. It is often used to connect devices to a computer. It operates in the ISM (Industrial, Scientific, Medical) band 2.4 GHz. In 2011, a Bluetooth Low Energy BLE (called Bluetooth Smart) variant was released as part of the Bluetooth 4.0 specification. Compared to previous versions, BLE consumes less power; has faster setup times; and supports an unlimited number of nodes in a star topology. BLE is specifically designed for low-bandwidth devices and devices using stand-alone power supplies. Launched to enable small scale consumer IoT applications in wearable fitness and medical devices (smart watches, glucometers, etc.) and smart home devices (door locks), through which data is easily communicated and viewed on smartphones [132, 133].

II.3.1.2 Near-Field Communication (NFC)

Near Field Communication (NFC) was developed by Sony and Philips for wireless communication. NFC uses the principle of magnetic coupling between devices to allow two-way data transfer when two devices are placed next to each other (maximum distance of 10 cm). NFC is based on RFID technology and operates up to 13.56 MHz. It contains a tag with a small amount of data and can be read-write when the device restores the data or read-only. The most common application is mobile devices for contactless payments. In other words, it stores and retrieves all kinds of personal data [134].

II.3.1.3 Wireless Fidelity (Wi-Fi)

Wi-Fi is considered the most widely used communication standard and is referred to as 802.11 by the Institute of Electrical and Electronics Engineers (IEEE). It is sponsored by the Wi-Fi Alliance, a non-profit organization that certifies products with Wi-Fi interoperability. Wi-Fi requires access points and wireless network adapters to set up a central Wi-Fi network. The Wi-Fi operates in the 2.4 to 5 GHz range. Today, Wi-Fi has become an important technology in the market due to the range and speed of data transmission.

Wi-Fi consumes a lot of power and is often not a viable solution for large battery-powered IoT sensor networks such as industrial IoT and smart buildings. Instead, it's used to connect devices that easily plug into an electrical outlet like smart home devices and surveillance cameras [135].

II.3.1.4 Cellular 3G, 4G, 5G LTE

A cellular network is a large area covered by radio waves divided into defined shapes and sizes called cells. Each zone has a transceiver (base station) that connects devices in the same cell or in different cells to other devices. Such networks offer low power consumption and wide coverage areas. Mobile networks first appeared in the 1990s as 1G (First Generation). Since then, new versions (2G, 3G, 4G LTE) have emerged, differing mainly in transmission type (analog-digital) and power consumption characteristics. 5G and LTE represent the latest generation of cellular technology and are popular for mobile phones and IoT devices that require high bandwidth. 5G networks will have higher speeds and more channels than previous generations, resulting in greater capacity and the ability to connect more devices [136].

II.3.1.5 LoRaWAN

LoRaWAN is a type of wireless network and one of the representatives of LPWAN (Low Power Wide Area Network) technology. LoRaWAN is often referred to in the literature by acronyms such as LoRa, LoRa Alliance, etc. LoRa (short for Long Range) is SemTech's physical layer and intellectual property that enables long-range communication. The Special Interest Group (SIG) has formed the LoRa Alliance of industrial and commercial partners who are co-developing an open spectrum network called LoRaWAN, a MAC layer over an OSI breadboard reference. It is the LoRa Alliance that claims that LoRaWAN is a communication protocol and network system structure, while LoRa is "only" the physical layer for long distance data exchange. LoRaWAN complies with European and North American regulatory standards. As announced by the LoRa Alliance in 2015, data transmission in Europe is in the 864 - 870 MHz ISM band (Industrial, Scientific and Medical), while in North America LoRaWAN also carries data in the 903-914.9 MHz band as well as the 923.3 - 927.5 MHz band [137]. Currently, many commercial and industrial solutions operate on LoRaWAN principles. It uses network and application servers to communicate with backend systems. It was called LoRa Network and Lora Application Server until November 2019. Since LoRa's name is protected by SemTech, the server was renamed ChirStack.

II.3.1.6 Narrow-band IoT (NB-IoT)

Narrow-band IoT (NB-IoT) is a network that belongs to the LTE-IoT wireless network type. It is based on the operation of 4G mobile networks. NB-IoT is cheaper than eMTC, so the main characteristics of this network are high coverage, lower power consumption than other LTE-based networks, and even cost. Wide coverage from 450 MHz to 3.5 GHz. NB-IoT will work on 2G and 3G networks, according to Qualcomm. The overall throughput of the NB-IoT network has been reduced to 200 kHz. NB-IoT networks operate on the principle of half-duplex communication and use TDD (Time Division Duplex) and FDD (Frequency Division Duplex). This divides the available band (into multiple frequency or time bands) and the transmitter or receiver uses the allocated bands to send and receive data. The first commercial NB-IoT product was developed by a working group responsible for adapting and accelerating NB-IoT systems, consisting of network providers Tele2, Telit, T-Mobile and the founders of Ericsson, Huawei and Nokia [138].

II.3.1.7 ZigBee

The network was born in the United States at the beginning of the 21st century. It was standardized by the IEEE in 2003. ZigBee is a wireless network that has established itself in the market due to its characteristics. While other wireless networks have tried over the years to introduce new features and improve performance, ZigBee continues to focus on 8-bit microcontrollers. The most famous feature of this network is the long life of the devices. Service life up to 10 years or more. ZigBee is similar to Bluetooth technology, but with features like lower data rates, transmission reliability, and affordability. It is not used to transfer large amounts of data, although some versions of the ZigBee protocol have been modified for such use. Due to its properties, it is widely used in sensor networks, and the transmission distance can reach 100 meters if there is good optical visibility [139] .

II.3.1.8 SigFox

Sigfox is an LPWAN network founded in 2009. It was the first private LPWAN network to enter the IoT market. There are two ways to manage your network. First, SigFox is the provider and second, SigFox has exclusive resellers. SigFox uses different frequency bands depending on your location. ISM 868 MHz in Europe, ISM 900 MHz in North America. SigFox has developed a long range communication system. Data transmission over long distances is "paid" by the data transmission speed. This is achieved by operating in the sub-GHz band and using UNB (Ultra Narrow Band) in combination with simple modulation techniques. Today, SigFox is present in about fifty countries around the world. Due to its popularity, some countries have invested in their network infrastructure to give their citizens access to the SigFox network (Belgium, for example) [140].

II.3.2 Types of IoT communication

Sensors or sensor data are at the heart of any IoT system. Sensors can be located anywhere in the world and data must be routed to a data center. There are many technologies for moving data to these data centers. This section provides an overview of these technologies.

The IoT combines Personal Area Networks (PAN), Local Area Networks (LAN) and long-range Wide Area Networks (WAN). Separate IP-based and non-IP-based communication systems. Non-IP-based communication systems are optimized in terms of cost and power consumption, while IP-based solutions such as 802.111 Wi-Fi are generally less constrained. Sensor data communication technologies can be divided into the following categories [141]:

1. Non-IP-based wireless personal area network (WPAN);
2. IP-based WPAN and wireless local area network (WLAN);
3. Long-range communication systems.

II.3.2.1 Non-IP-based Wireless Personal Area Network

Internet-connected sensors need a way to send and receive information. This is the subject of Personal Area Network (PAN) and short range communications. Communication with

sensors or actuators can be the widely used chopper or wireless personal area network (WPAN). Wired connections are still used, but mainly in industries and regions that are not compatible with radio frequencies. Here are some WPAN standards:

— 802.15 standards;

— Bluetooth;

— ZigBee;

— Z-Wave.

The Bluetooth, ZigBee, and Z-Wave protocol stacks are similar to the actual TCP/IP protocol, but they do not inherently communicate over TCP/IP.

II.3.2.2 IP-based Wireless Personal Area Network

Supporting the IP layer in the protocol stack consumes resources that could be used elsewhere. However, creating an IoT system that allows devices to communicate over TCP/IP has distinct advantages. Regardless of the protocol used at the sensor level, sensor data eventually ends up in public, private, or hybrid clouds for analysis, control, and monitoring.

IP is the standard form of global communication for several reasons: ubiquity, durability, scalability, reliability, and manageability.

Some WPAN IP-based standards:

— WPAN with IP, 6LoWPAN;

— WPAN with IP, Thread;

— IEEE 802.11 protocols and WLAN.

WPAN with IP, 6LoWPAN

The 6LoWPAN concept was developed in 2005 to bring IP addressing capabilities to the smallest and most resource-constrained devices. 6LoWPAN stands for IPv6 over Low Power WPAN. The Internet is intended for IP networks over low-power communication systems for devices that are limited in power and space and do not require high-bandwidth network services. 6LoWPAN is a mesh network located at the edge of large networks [142].

WPAN with IP, Thread

Threads is a relatively new network protocol and is based on IPv6. Its main objectives are home connectivity and home automation. It is based on the IEEE 802.15.4 and 6LoWPAN protocols and has some things in common with ZigBee and other IEEE 802.15.4 variants, but the main difference is that the Threads are IP addressable.

Thread are also lattice-based, making them attractive for home lighting systems. Thread's philosophy is that enabling IP addressing on smaller sensors and home automation systems can reduce power consumption [143].

II.3.2.3 Long-range communication systems

Let's look at a few wide area network (WAN) technologies. The two WAN technologies are 4G-LTE and 5G cellular, with their own systems including long range radio (LoRa) and Sigfox.

4G-LTE and 5G

The most common form of communication is cellular radio communication. Mobile communication devices existed for many years before cellular technology, but they were essentially two-way radios because they had a limited range and shared a common frequency space. A typical 4G LTE network consists of three components: client, wireless network, and core network. A radio network represents the front-end communication between clients and the core network and includes radios such as towers.

5G is the next-generation IP communications standard that will replace 4G-LTE. 5G promises significant capabilities for IoT and commercial, mobile and vehicle use cases. 5G will also improve bandwidth, latency, density and cost of use [144].

LoRa and LoRaWAN

LoRa is the physical layer of a long-range, low-power IoT protocol, and LoRaWAN is the MAC layer that moves data packets between network interface cards. LoRa represents the physical layer of the LoRaWAN network. Manage modulation, power, receiver and transmit radios, and signal conditioning. LoRaWAN is based on a star network topology and relies on cloud-based network interfaces [145].

Sigfox

Sigfox is a narrow-band LPWAN protocol developed in Toulouse, France in 2009. Originally, Sigfox was designed as a pure unidirectional sensor network. This meant that only communication from the sensor uplink was supported. Since then, down-link channels have become available. A Sigfox network can have a density of up to 1 million nodes per base station. The density is a function of the number of messages sent by the network structure. All nodes connected to the base station form a star network [146].

II.3.3 Some application protocols in IoT

We call the set of rules that define a mode of communication between two computer applications an application protocol.They rely on the TCP and UDP transport protocols to first route and exchange data according to all the rules of the chosen application protocol.

For raw data to be received by the IoT platform, it needs a "facade" that objects can connect to and communicate with. This facade represents the so-called application interface, where the application protocols are used.

Note that traditional web infrastructure is not suitable for most IoT applications. Indeed, some connected objects, called constrained, are limited by low power microcontrollers and having small amounts of memory (flash and RAM), while the limits on IoT networks, in particular ZigBee, due to the error rates high and low bit rate packets (a few tens of

kbit/s). Therefore, a less verbose protocol with a limited number of messages and a small size is needed to overcome these limitations.

We have identified three families of protocols and selected the most commonly used application protocols for each:

— Messaging protocol: MQTT, XMPP and AMQP.

— Web Transfer Protocol: CoAP, REST API.

— Network protocol: Websocket.

II.3.3.1 Messaging protocols

The messaging protocol is based on a publish and subscribe mechanism in which data transfer is asynchronous. Let's present the operation and characteristics of the MQTT, XMPP, AMQP protocols [147].

Message Queue Telemetry Transport (MQTT)

MQTT (short for *Message Queuing Telemetry Transport*) is a publish/subscribe messaging protocol based on the TCP/IP protocol. A client named *publisher* first establishes a 'publish' type connection with an MQTT server named *broker*. Publishers then forward messages to brokers on specific channels called *topic*. These messages are read by members called *subscribers* who have previously established a subscription-type connection with the broker. Sending and consuming messages is therefore an asynchronous [148]. The operation that has just been described is illustrated in the diagram below (Figure II.4).

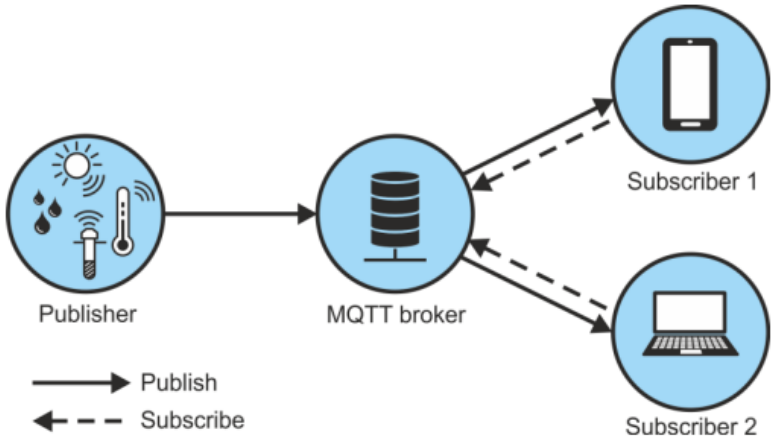

Figure II.4: How the MQTT protocol works [148].

MQTT is characterized as a lightweight protocol due to the limited and small number of messages. Each message actually consists of a 2-byte fixed header (indicating the type of message and the quality of service level used), an optional variable header and a payload limited to 256 MB. The way that the MQTT protocol handles content is determined by referring to three levels of quality of service (QoS). Subscribers can specify the maximum level of QoS they wish to receive. However, higher quality levels result in higher latency and bandwidth due to additional retries and acknowledgments [149, 150].

In summary, the properties of the MQTT protocol meet the following requirements,

making it a suitable protocol for IoT networks:

- Very suitable for low bandwidth networks;
- The limited number of small messages makes it ideal for use on wireless networks;
- Low energy consumption due to fast publication and consumption of messages;
- Requires very few compute and storage resources;
- Send messages to multiple entities over a single TCP connection.

Extensible Messaging and Presence Protocol (XMPP)

Originally the instant messaging protocol XMPP, for *Extensible Messaging and Presence Protocol*, is used specifically by the Jabber and Google Talk services. Its scalability has allowed its use in other applications such as VoIP. Its operation is based on a client/server architecture and the exchange of data in XML format follows the same principles as electronic messaging. Indeed, the communication between the two clients is asynchronous and goes through the XMPP server. First, the client establishes a TCP connection with the XMPP server and sends data to the recipient's XMPP server. The latter will send the data to the receiver if the receiver is connected. Otherwise, the XMPP server stores the data until each recipient is connected. We can add that an XMPP system is decentralized and potentially real-time if the sender and receiver are connected during message delivery. In addition, each client is identified by a unique identifier based on the following pattern: <client-name> @ <server-name> [151]. The following diagram (II.5) illustrates the principle. When client A sends a message to client E, client E specifies in the XMPP message:

— The identifier of the issuer (here clientE@domainA)

— The recipient's identifier (here clientE@domainB)

— Information conveyed in the body part (here Hello IoT).

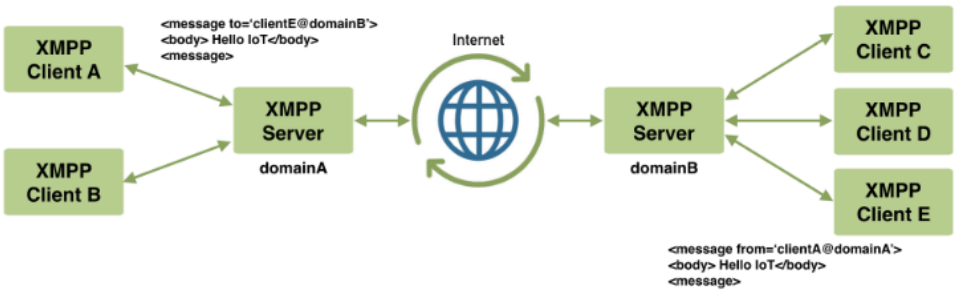

Figure II.5: How the XMPP protocol works.

The main advantages of this protocol are: addressing using a unique identifier; simple security settings; a message format providing structured data and a server system. All of this makes it a suitable protocol for M2M (Machine-to-Machine) applications, as opposed to the MQTT protocol. Indeed, it better manages the integration of new connected

objects, thus allowing interoperability with other IoT platforms and therefore with other ecosystems.

Advanced Message Queuing Protocol (AMQP)

AMQP, for *Advanced Message Queuing Protocol*, is a messaging protocol that represents an alternative solution to the commercial products MOM (Message-Oriented Middleware) and JMS(Java Message Service) [152]. AMQP has been in development since 2003, but this protocol is an exception because it is not the usual initiative of IT companies. In fact, it was first developed by an American bank, JPMorgan Chase.

The AMQP protocol works on the same principles as MQTT, but the publisher/subscriber concept is replaced by the producer/consumer concept [153]. Moreover, thanks to an internal mechanism called "Exchange", AMQP makes it possible to forward messages from a producer to several topics. Routing criteria can be defined in different ways: Inspection of content, headers, routing keys, etc. Therefore, the same message can be consumed by different consumers across multiple topics [154, 152].

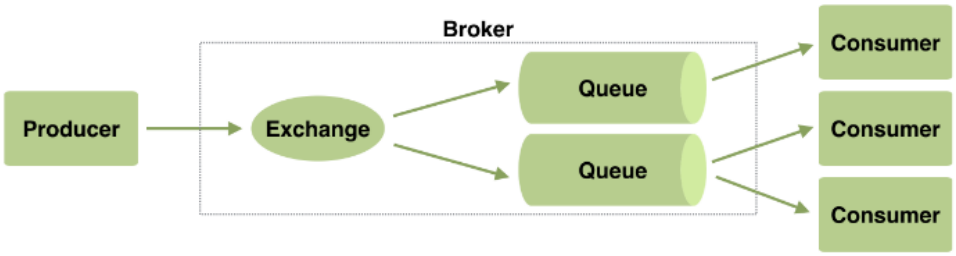

Figure II.6: How the AMQP protocol works.

AMQP is therefore better suited to situations where reliability, more advanced messaging scenarios, interoperability between protocol implementations, and security are needed. It is therefore intended for networked objects with low communication constraints and high security requirements.

II.3.3.2 Web Transfer Protocols

Constrained Application Protocol (CoAP)

CoAP, for *Constrained Application Protocol*, was developed by the IETF Constrained RESTful Environments (CoRE) group as an application layer protocol for IoT systems. CoAP is a web protocol based on a client/server architecture. CoAP introduces the Web REpresentational State Transfer (REST) protocol in addition to HTTP. Unlike REST, CoAP uses lightweight UDP as the default transport protocol (TCP is not supported). This protocol includes some methods and nomenclatures from the HTTP protocol. On the other hand, unlike the HTTP protocol which is based on the TCP/IP suite, the CoAP protocol is based on the UDP/IPv6/6LoWPAN suite, greatly simplifying the message exchange mechanism defined by the UDP protocol [155, 156].

Four types of messages are used by CoAP: confirmable, non-confirmable, reset, and acknowledgement. CoAP achieved its reliability by using confirmable and unconfirmable messages. Similar to HTTP, methods like GET, PUT, POST, and DELETE are used by

CoAP to perform create, retrieve, update, and delete operations [157].

REST API

REST, for *REpresentational State Transfer*, is a protocol that allows the management, identification and manipulation of resources (users, images, sensor data, etc.) through an application programming interface (called API for *Application Programming Interface*). This interface corresponds to a set of URIs accessible via different HTTP request methods (GET, PUT, POST and DELETE). The server response contained in the body of the HTTP frame can be delivered in several formats. XML and CSV formats are possible, but JSON is often the preferred one [158]. The server also adds a three-digit HTTP response code to indicate response status, it has the following form [159]:

- 2xx: Indicates success of client request processing (example: 200 for OK).
- 3xx: Redirect the client to another link.
- 4xx: Indicates that there is an error in the client's request (example: 404 for Not Found).
- 5xx: Suggests a server error (example: 500 for Internal Server Error).

REST APIs are widely used and have proven to be easy to configure and modify through a combination of methods and URIs. It is suitable for connected objects without communication constraints, for example, smartphones. Figure II.7 shows an example of how the REST API works.

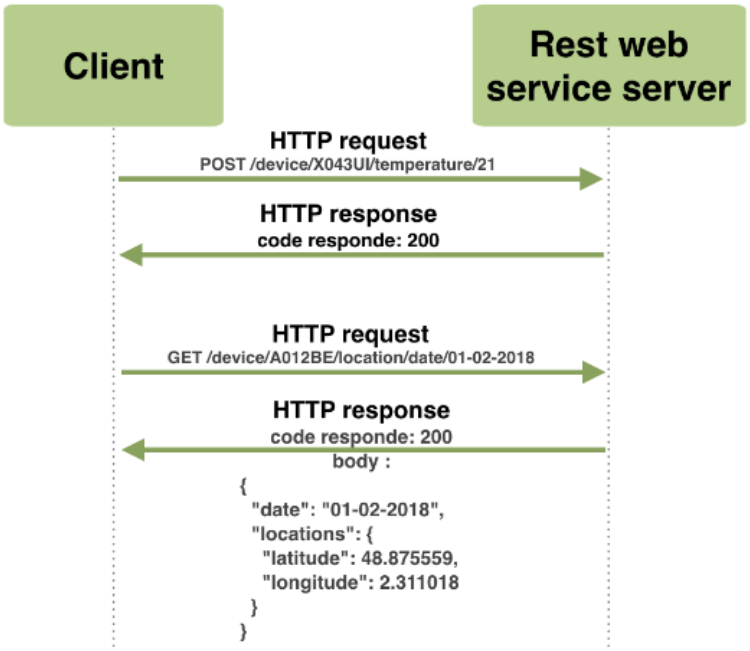

Figure II.7: Example of how an REST API works.

II.3.3.3 Network protocol (Websocket)

The Websocket protocol allows a single TCP connection between a client and a server to establish a full-duplex communication channel. The diagram II.8 shows the three main stages in the life of the channel:

1 - The client initiates a connection phase called "handshake".

2 - Two-way messaging phase.

3 - Canal closing phase initiated by either party.

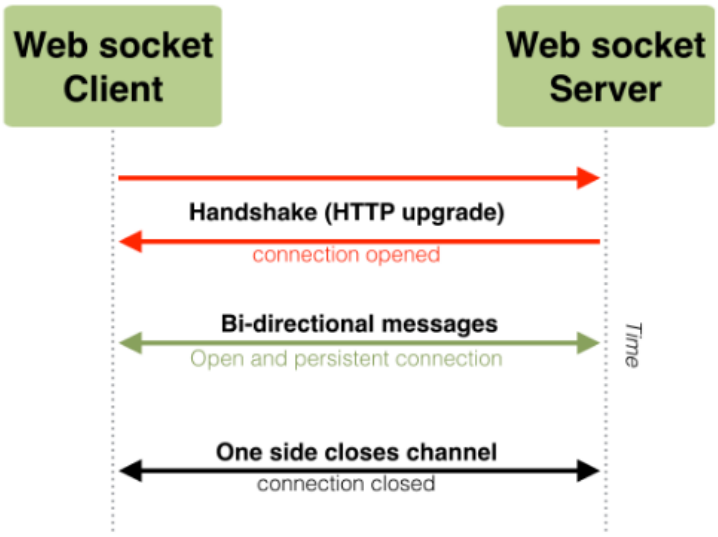

Figure II.8: How a Websocket protocol works.

This protocol was originally implemented to overcome the shortcomings of the HTTP protocol by providing two-way communication between web applications and server processes. Communication is actually asynchronous. In other words, the server can only send a response if the client has already sent a request. Websockets therefore allow the exchange of messages in real time, which is ideal for alerts and notifications [160, 161].

Websocket is a type of communication that is expensive in terms of resources, whether hardware (CPU, memory, power) or bandwidth, so it is only suitable for sensors with sufficient resources. So, It is mainly suitable for monitoring the situation and transmitting information in real time.

II.4 Towards the Web of Things

Participating objects in an IoT network are heterogeneous in several aspects, hardware, software, technologies and communications protocols. Therefore, heterogeneity is one of the essential characteristics of the IoT system, which it must face, because the IoT ecosystem requires connecting and integrating heterogeneous objects and systems. To overcome this challenge, the first step is to ensure inter-connectivity of the different devices, i.e. to connect them to the Internet. This connectivity is implemented by integrating all the devices in a common address space. IPv6 addressing, which makes it possible

to connect all objects in a homogeneous way [162]. However, the overhead due to IPv6 headers must be taken into account since these devices have few resources. To this end, several solutions have been proposed, to allow the use of IPv6 while preserving resources, such as 6LoWPAN [163] and IPv6 Addressing Proxies [164].

Once the connectivity is achieved, a common transport and application layer protocol is required. The most widespread solution in the Internet is undoubtedly the Web, thus creating what is called *Web of Things* (WoT) [162]. It is an enhancement of the Internet of Things that integrates smart objects, not only in the Internet, but also on the Web (i.e. at the application layer) [165]. Embedding objects in the web makes them accessible as resources like any other web resource and provides remote access using web applications.

II.4.1 Integrating Objects into the Web

To integrate objects into the Web, two solutions are possible, direct integration and indirect integration (through a proxy) [165, 166, 167].

II.4.1.1 Direct integration

To integrate objects directly into the Web, first of all, they must be addressable, i.e. that they may have an IP address, or be compatible with the IP protocol when connected to the internet. The WoT also requires connectivity and interoperability at the application layer. For this reason, a Web server must be integrated in such a way that objects can understand each other through languages and protocols specified by Web standards [166]. The direct integration can be illustrated in the Figure II.9.

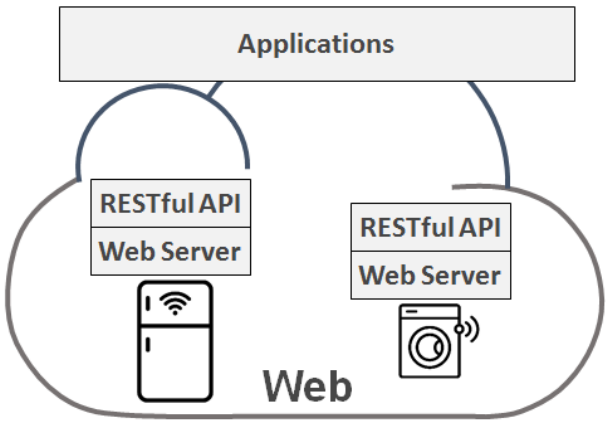

Figure II.9: Direct integration of objects in the Web [167]

II.4.1.2 Indirect integration

Indirect integration is adopted in situations where direct integration of a Web server on an object is not feasible or desirable (too limited resources, additional cost, security reasons, etc.). In this case, the web integration is done using an intermediate proxy that connects these objects with the web, this type of proxy is called Smart Gateway, it is a network component which does more than transmit data. An Intelligent Gateway is in fact a Web

server which summarizes the communication (eg Bluetooth or Zigbee) between objects and the Web through the use of dedicated drivers [165]. Figure II.10 illustrates indirect integration.

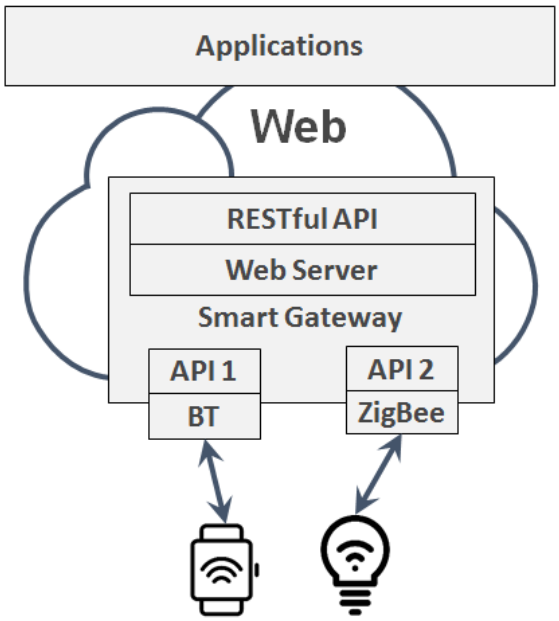

Figure II.10: Indirect integration of objects in the Web [167]

II.4.2 WoT architecture recommended by W3C

Motivated by the lack of interoperability between different application domains and IoT devices, the *World Wide Web Consortium* (W3C) focused on defining a vendor- and application-agnostic platform to foster interoperability between the various IoT devices. Such a platform is called *W3C Web of Things (WoT) Architecture* which allows IoT devices to communicate with each other regardless of their specific implementation [168].

Overall, the W3C WoT architecture builds on well-known existing web standards and provides a standard method for describing IoT device interfaces. Figure II.11 illustrates the different components defined in the WoT architecture. Each device must provide a metadata file called *Thing Description* (TD). This metadata is self-describing, so users can identify what capabilities an object provides and how to use those capabilities [169, 170].

A TD provides a semantic description of a specific IoT device (i.e. it describes an individual object, not a type of objects), along with its interactions. In other words, the TD defines the semantics and how to interact with a specific IoT device.

The interaction types of a TD are *Property*, *Action* and *Event*. These interactions cover the majority of interactions with an IoT device. Properties describe attributes or state of an IoT device (e.g. humidity value, door is locked or unlocked), actions invoke processes (e.g. turn on a light, open a window), while events notify asynchronously of the occurrence of a specific condition (e.g. alarm activated, lamp overheated).

TD relies on RDF as the underlying data model. Additionally, TD instances are serialized in JSON-LD format. JSON-LD adds a semantic layer on top of the JSON

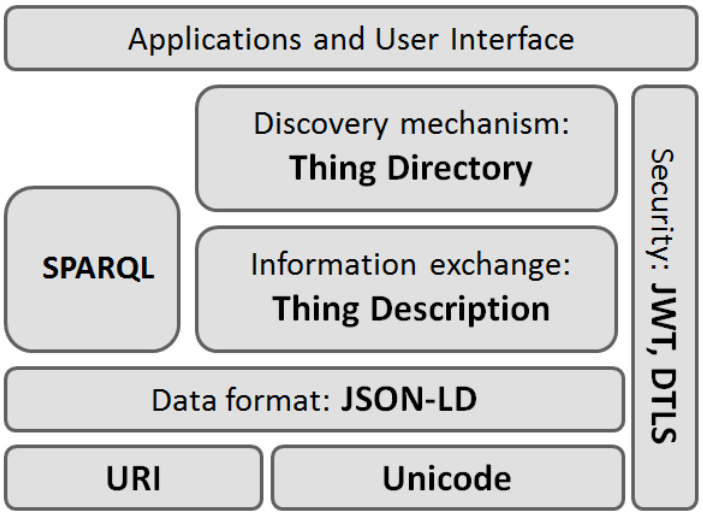

Figure II.11: W3C WoT Architecture [170]

specification, which means that terms that appear in a TD are associated with concepts from a common vocabulary.

The listing II.1 illustrates the content of a TD document, which are:

- **A property named *"status"*:** this property is accessible via the CoAP protocol with a **GET** method at the URI *coap://mylamp.example.com/status*, and returns a character string value (lines 4 to 8).
- **An action is specified *"toggle"*:** to toggle the switch state using the **POST** method on the URI *coap://mylamp.example.com/toggle* (lines 9-13).
- **An event *"overheating"*:** allows you to set up a mechanism for sending notifications by the object. Here, in this example, a subscriber will be notified of a possible lamp overheat on the URI *coap://mylamp.example.com/oh (lines 14-20)*.

TD instances can be placed in a component called *Thing Directory*. The Thing Directory stores TDs and provides an interface for registering, updating, and deleting TDs [171]. Such an interface also allows performing search operations based on SPARQL queries. The latter is a semantic query language capable of retrieving and manipulating data stored in RDF format [172].

Once a device obtains the TD of another IoT device from the Thing Directory, it must first interpret and understand the information it contains to interact with it. Subsequently, communication between the devices can take place automatically, without manual intervention.

```
{   "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "properties": {
        "status" : {
            "type": "string",
            "forms": [{
                "href": "coap://mylamp.example.com/status",
                "methodName" : "GET"}]}},
    "actions": {
        "toggle" : {
            "forms": [{
                        "href": "coap://mylamp.example.com/toggle",
                        "methodName" : "POST"}]}},
    "events":{
        "overheating":{
            "data": {"type": "string"},
            "forms": [{
                "href": "coaps://mylamp.example.com/oh",
                "methodName" : "GET",
                "subprotocol" : "observe"}]}}}
```

Listing II.1: Content of a TD instance .

The WoT provides the ability to interact with physical objects as a web service, and exposes remote access using web applications. The majority of such applications use *Service Oriented Architecture* (SOA) implemented by Web services to host services offered by real-world objects (IoT devices). The following subsections introduce the SOA concept, web services and discuss their applicability to the WoT as well as their contributions for IoT interoperability.

II.4.3 Service Oriented Architecture

The term "SOA" is used to refer to the service-oriented approach as a whole. The SOA approach aims as a first objective to ensure interoperability between heterogeneous systems, which can be used to design infrastructures allowing customers and suppliers to exchange services, despite the disparity of domains, technologies and suppliers [173]. This approach provide loose coupling between different systems and promotes re-usability and interoperability.

In the context of the Internet of Things, an object can be both a consumer and a service provider. This consideration is due to the fact that IoT interactions are between machines (M2M). It is the input and output flows that define the roles of objects at a given time between consumer and supplier. Figure II.12 presents the SOA architecture.

The architecture of Figure II.12 allows a service provider to describe its service and publish it in a directory to make it available to users. A client (user or service consumer) initiates a service discovery operation by searching the directory. Once found, the customer begins using the service.

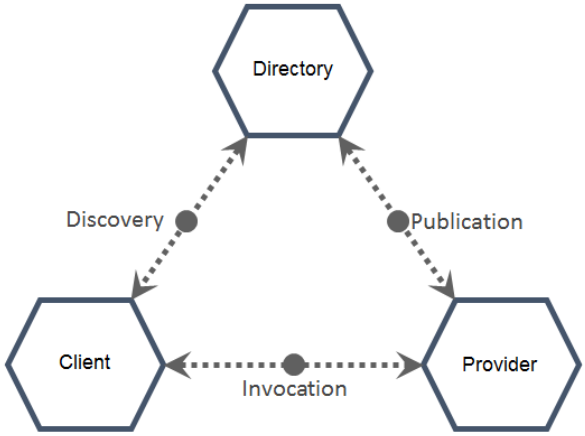

Figure II.12: Service Oriented Architecture.

II.4.3.1 Web service

According to the W3C, a web service is a software system designed to support interoperable machine-to-machine interaction over a network, through an interface described in a machine-readable format following the SOA architecture [174].

II.4.3.2 Implementing the SOA architecture

There are two approaches to implementing the SOA architecture, based on web services, standard web services (WS -*) and REST web services.

WS -* approach

In standard web services (WS -*), the functions of description, publication, discovery and invocation of services must be performed using standard protocols developed specifically for these tasks, as follows [175]:

- **Description:** To describe a Web service, the language WSDL (Web Services Description Language), based on XML, is used, specifying the available methods, the formats of the input and output messages, and how to access them;.
- **Publication:** Consists of registering the Web service description in a directory called UDDI (Universal Description, Discovery, and Integration);
- **Discovery:** Find the description of the desired web service in the UDDI directory;
- **Invocation:** The interaction between the consumer and the provider takes place using the SOAP protocol (Simple Object Access Protocol), based on XML, and relies on a transport protocol (HTTP, SMTP, POP etc.).

REST approach

These web services are based on a style of architecture called REST (REpresentational State Transfer) [176], this architecture allows to build different types of web applications,

called RESTful, based on a set of conventions and good practices to be respected and not a standard or a technology in its own right.

The REST architecture uses the Web as the application platform and fully exploits the functionalities inherited from the HTTP protocol, rather than adding an overlay. Therefore, it offers faster communication, directly using some methods defined in HTTP, such as POST, GET, PUT and DELETE. Another advantage of REST Web services is that they allow the use of several resource representations, in different formats (XML, JSON, CSV, Text/plain, etc.) and not only in XML, which greatly reduces the messages to be transferred and the communication will then become faster.

The WoT adapts the REST approach in order to realize the application layer of the IoT. However, given the complexities of the HTTP protocol for constrained objects, a new RESTful application protocol called CoAP (Constrained Application Protocol) has been designed [177]. CoAP is a RESTful web transfer protocol for M2M communications in constrained networks. The CoAP specification defines a client/server model, similar to HTTP, where a CoAP endpoint can typically act as both a server or a client. As CoAP is intended for constrained networks, it introduces low header overhead and reduces analysis complexity. Some of the features of the CoAP include:

- Unicast and multicast support;
- Transactions with and without acknowledgment of receipt;
- Four different methods, similar to those of the HTTP protocol:
  - GET (retrieving the representation of a resource);
  - POST (create a new resource);
  - PUT (update a resource);
  - DELETE (deleting a resource).
- Three types for response codes:
  - 2.xx (success);
  - 4.xx (customer error);
  - 5.xx (server error).

By using CoAP, all objects in an IoT environment can provide RESTful applications, thus enabling the construction of an easily accessible Web of Things.

With the web of objects, problems of heterogeneity between objects can be solved. Nevertheless, being syntactic, the resources are structured and well defined, but the content remains difficult to exploit by machines. To overcome this, it is necessary to provide semantics in the various IoT applications, in order to be able to make the best use of the available resources in order to provide relevant answers.

II.4.4 Semantic Web of Objects

The WoT is moving the IoT towards a common stack based on web services. However, even when seamless access is achieved through web protocols, we still need semantic interoperability, i.e. the ability of devices to unambiguously convey the meaning of the data they communicate. For this, the technologies of the Semantic Web is adopted by the Web of Things, thus generating what is called the *Semantic Web of Things* (SWoT) [178, 170].

The objective of SWoT is to implement systems based on knowledge representation and reasoning techniques that achieve a high degree of WoT autonomy thanks to the capabilities of the Semantic Web.

II.5 Conclusion

In this chapter, we have introduced IoT basics, features and applications. Then we focused on the issues and challenges that are slowing down the progress of the IoT, of which heterogeneity is a part. After that, we reviewed the main communication technologies and protocols governing the field. We also discussed the notions of WoT, SOA architecture and SWoT, and how they can overcome the heterogeneity problem by promoting interoperability between different IoT devices regardless of their specific implementation. The thing that will facilitate the discovery and selection of IoT services, and make the *IoT services composition* possible. IoT Services composition consists in creating a new IoT service by combining existing atomic ones in order to meet a complex need.

Chapter III

IoT services composition, State of the art

III.1 Introduction

The Internet of Things (IoT) is a paradigm characterizing physical objects each having their own digital identity and capable of communicating with each other. We saw in the previous chapter, how WoT and SOA architecture technologies can overcome the problem of heterogeneity between connected objects and different platforms by making them interconnectable. On the odder hand, the loose coupling between IoT services promotes the reuse and composability. In fact, service composition consists in creating new services by combining existing atomic ones, taking into account the user needs and the context of the environment.

In the first part of this chapter, we present the fundamental concepts of IoT service and its properties. By emphasizing the non-functional aspect, in particular Quality of Service (QoS) properties. Then, we present the services composition process, its steps, of which the service selection represents a necessary and important step, as well as the different services composition types. The second part will be devoted to a review of the literature, in which we will study a panel of proposed approaches to solve the problem of QoS-aware IoT services composition, with a comparative study.

III.2 Service, Service composition

III.2.1 Service representation

III.2.1.1 Definition

A service is a software entity making it possible to expose one or more functionalities defined by an interface. This interface contains the functional properties of the service, that is to say, the action that the service must perform and, possibly, non-functional properties allowing to identify the service which best suits the needs of the users [173].

III.2.1.2 Concrete services

A concrete service is a function that acts on input data to provide output results. It is described by functional and non-functional properties. The functional properties include the service input data which will be transformed into output data after the execution of the service. While, non-functional properties are defined mainly by a set of contextual information to which the service is sensitive e.g., location, quality of service (QoS) attributes [179].

III.2.1.3 Abstract services

An abstract service is a class of concrete services that are functionally equivalent, i.e., have similar functionality. Two concrete services are said to be functionally equivalent, if the two services have the same input data and the same output data, but differ in their non-functional properties [179]. An abstract service is considered to be a class of concrete services that is both abstract and dynamic. On the one hand, an abstract service is seen as an abstract class because it does not have a single real implementation. On the other hand, an abstract service is seen as a dynamic class because its real implementation depends on the concrete service chosen during the execution phase.

III.2.1.4 Service properties

The identity of a service is defined by its functional and non-functional properties. Indeed, the functional properties are described in its description in terms of operations, and reflect the functioning of the service. On the other hand, the non-functional properties of a service represent a set of quality of service (QoS) attributes. These attributes represent the ability of the service to respond adequately and under the right conditions to the user's requirements in order to satisfy him [180].

Functional properties

Functional properties reflect the behavior of a service described from an operational perspective. The latter is generally described by his two kinds of information, the "Inputs" set and the "Outputs" set. Inputs describe the data required for a service operation, and outputs describe the data produced after its execution. Other semantic extensions can be used to complete the functional description of the service, including preconditions and postconditions. Preconditions describe the state of the world required for service execution, and postconditions describe the impact of service execution on the world [179].

Non-functional properties (QoS Attributes)

A service's non-functional properties (also known as *Quality of Service* (QoS)) allow you to evaluate how the service behaves or performs from the perspective of service providers and users. These parameters correspond to observable or computable metrics. Quality of Service (QoS) can refer to non-functional properties such as availability, reliability, and security, etc. Each property is called *QoS attribute* and gives information about a certain quality value provided by the IoT Service. QoS attributes can be divided into two categories: (1) *qualitative attributes* (security and privacy); (2) *quantitative attributes* (cost and response time) [179, 181, 182].

QoS attributes can also be divided into two classes according to their impact on the overall IoT service QoS: (1) *positive attributes* (availability, throughput, etc.) which have a positive impact, as this QoS attribute value increases, the QoS of the IoT service increases as well; (2) *negative attributes* that have a negative impact (execution time, cost, etc.), decreasing the value of this kind of attribute will increase the QoS of the IoT Service. In the context of service composition, the positive QoS attributes of a composite service should be maximized whereas the negative attributes should be minimized [179, 183].

The QoS attributes of a concrete IoT service are usually represented by a vector of dimension q, where q represents the number of QoS attributes. The most commonly used quality of service (QoS) attributes in the literature are described in Table III.1.

QoS Attribute	Description
Throughput	It represents the number of requests that the IoT Service can handle in a given time interval. This can include a maximum throughput, or a function that shows how throughput varies with the intensity of the load.
Response time	It indicates how long the IoT service should respond to a particular request. This period may include processing time and sending time.
Availability	Represents the probability that the IoT service is active at the time it is invoked.
Reliability	It refers to the ability of an IoT service to perform its functions correctly over a period of time. This can be measured by the average time between two failures.
Security	IoT service security includes various aspects that ensure that message exchanges between users and services are secure. More specifically, IoT service security includes several aspects such as confidentiality, message encryption, and access control.
Reputation	Reputation is a measure of the credibility of a service. This mainly depends on the experience of the end users who have already accessed the service.

Table III.1: The most used QoS attributes in IoT services

III.2.2 Service composition

Service composition provides user-requested functionality by combining the functionality of several atomic services. Service composition can be static or dynamic. A static composition is realized at design time while a dynamic composition is realized at run-time. Several definitions of service composition have been proposed in the literature. In what follows, we will summarize the most significant ones.

III.2.2.1 Definition of the service composition

Service composition is defined as a process from the discovery of services to their composition to meet a complex user request. In [184], Service composition is defined as the ability

to form new value-added services by combining existing ones that are probably offered by different service providers. In [185], service composition is considered an efficient way to create, execute and maintain services that depend on other services. Service composition can be an aggregation and combination of a set of services to achieve a common goal [186]. In fact, the IoT composite services enables the aggregation of intelligent object services to meet complex needs [187].

III.2.2.2 Types of service composition

Service composition approaches can be classified according to two main aspects. The first concerns the degree of user participation in defining the composition plan, and in this case composition can be *manual*, *semi-automatic* or *automatic*. The second concerns the management of services, in this case, the composition will be called *static* or *dynamic*.

Composition approaches Static Vs. Dynamic

In static approaches, the combination of IoT services is realized during the design phase of the composite service. The IoT services are linked together before being deployed. In this type of composition, the substitution or modification of a service requires changes in the composite service design. Therefore, this type of composition is more suitable for environments that are not subject to rapid changes over time. Unlike static composition, dynamic composition allows the combination of IoT services at the time of the request execution. Dynamic composition allows to select and combine dynamically the services from the directories in which these services are published while taking into account the user's constraints [188].

Composition approaches Manual Vs. Semi-automatic or Automatic

A composite service offers more complex functions than those provided by an atomic service. However, as the number of services increases and user requirements become more and more complex, the task of finding the most suitable services for the user's requirements becomes more and more complex. Also, it is becoming more and more difficult to build a composite service manually. Manual composition relies on the intervention of the user without the intervention of specialized tools; the user will use a simple text editor to program all the composition steps [189]. This type of composition requires the user to have a developer profile. The composition then becomes a tedious task requiring efforts to be provided by the user. As opposed to manual composition, in automatic composition, user intervention is required only for the specification of requirements in the form of an abstract request. Thus, the composition process is carried out in a transparent way for the user. The semi-automatic composition approach is a compromise between the manual approach and the automatic approach. It allows the user to keep a certain control and to supervise the composition process without having a programmer's profile by relying on the assistance provided by graphical tools. [190].

III.2.2.3 Steps in the service composition process

Service composition is the process of combining atomic services. Figure III.1 shows the steps of composition including its four phases : (1) design of composition plan, (2) services discovery, (3) services selection, and (4) execution and control of the composition [179, 190].

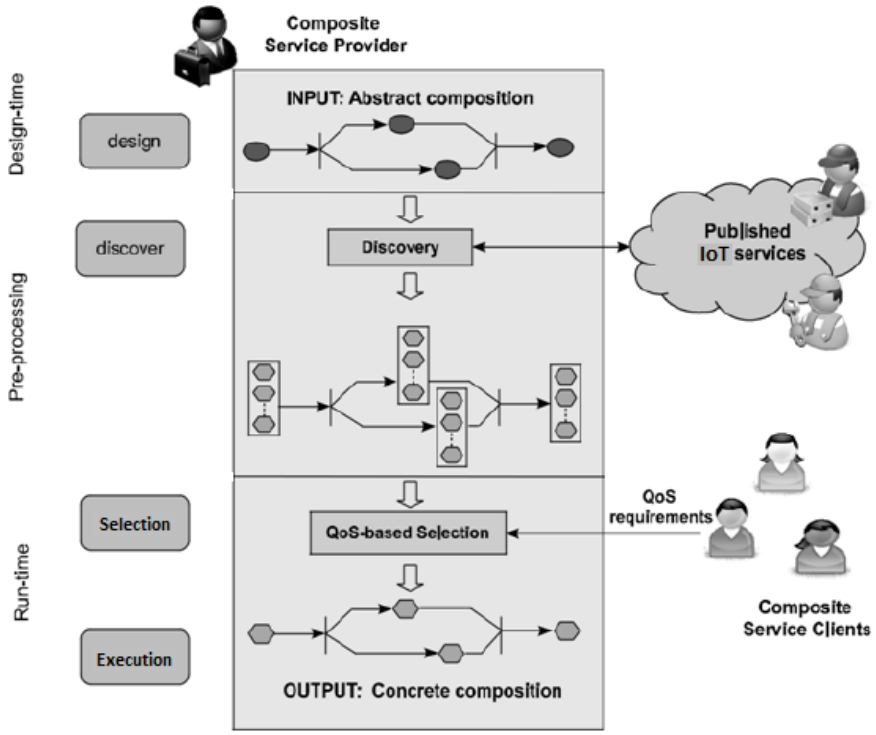

Figure III.1: Service composition steps [179].

1. **Design of the composition plan:** In this phase, the service requestor specifies a service composition request. This request should provide sufficient information about the user's requirements and preferences for the composite service. This phase identifies the functional and non-functional needs of the service resulting from the composition. These requirements can be specified as an abstract description containing a set of abstract services, each representing a different function.

2. **Services discovery:** This phase identifies concrete services that meet the functional requirements defined in the design phase. A search within the service directory can discover a set of concrete candidate services with similar functionalities, with different QoS values for each abstract service in the composition plan.

3. **Services selection:** This phase consists of determining the most suitable concrete service for the composition from the set of candidate services obtained in the discovery phase. Once all required IoT services have been identified and linked to their associated activities, a concrete composition is created.

4. **Execution and control of the composition:** During this phase, a composite service instance is created and executed by an execution engine that is also responsible for calling individual service components. Monitoring tasks such as logging, execution tracing, performance measurement, and exception handling need to be performed while a composite service instance is running.

III.2.2.4 Services selection, an important step in service composition

Service selection is an important step in service composition. It refers to the question: how to efficiently choose services among a set of functionally equivalent services, on the basis of functional and non-functional properties? The selection process begins with a request from the user in the form of a composite abstract service. The discovery module uses the service directory to find the services available for each abstract service in the abstract plan provided by the user, performing a functional correspondence between the abstract services and the descriptions of the available services. Therefore, a list of concrete services is obtained for each abstract service. The objective of QoS-aware service selection is to choose a concrete service from each abstract service in such a way as to have a QoS that satisfies the user's requirements, as shown in Figure III.2.

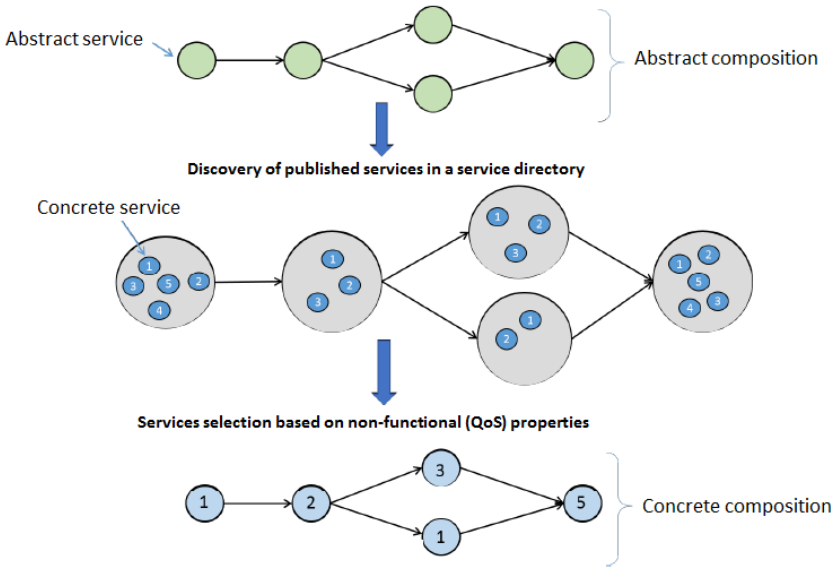

Figure III.2: IoT service selection process.

Two types of service selection exist, *local selection* and *global optimization*:

- **Local selection**

  The local selection of services consists in finding for each request, independently of the other requests, the most appropriate services. This type of selection makes it possible to take into account detailed QoS requirements at the level of each service of the composition (local QoS constraints), but it does not guarantee obtaining a composite service satisfying the global requirements in terms of QoS (global QoS constraints) imposed by the user.

- **Global optimization**

  Global optimization tends to solve the limitation of local selection in terms of taking into account global QoS constraints. It evaluates the composite services to determine the best composite service in terms of QoS and which satisfies the overall QoS constraints. However, this type of selection generates a significant execution time.

III.2.3 SOA: service choreography and orchestration

In this section, we will define, in a global way, what are the two approaches of the applications architecture built around the composition of services. Then we will look at the advantages and disadvantages of these *Service Oriented Architectures (SOA)* [191, 192].

III.2.3.1 Orchestration: centralized processing

The first approach to service collaboration is based on a concept of centralizing control of application flow. Inherited from structured development methods, service orchestration addresses the problem of using the processing power of remote and interconnected machines by reproducing the practices of this mode of programming, such as function calls, parameters , return values, etc. In this case, the main program, executed on the central node, retains complete control of the running of the algorithm. It collaborates with the services in synchronous mode whose waiting for the service response blocks the program execution. It then retrieves the result of this call, integrates it into its own management and continues its progress.

In this type of organization, only the central node has control of the application. This does not prevent the services used from having their own logical progression. However, the central point dominates the other participants, because it is the only one with the complete view of the business logic and its progress. The dependency is clearly oriented, the participating services not having the possibility of reversing the direction of the decision-making. Figure III.3 illustrates the service orchestration approach.

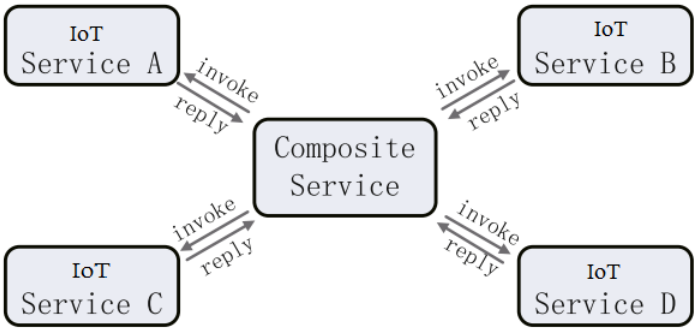

Figure III.3: IoT Service orchestration.

III.2.3.2 Choreography: Distributed Processing

Some applications sometimes depend less directly on a central element. In some cases, the responsibility for the progress of all the reactions can be distributed over several control points, each of which can in turn take on the role of decision-maker and thus pilot the progress of the exchanges. In these cases, the centralization imposed by the orchestration turns out to be unsuitable. The traditional image of the "orchestra leader" who directs an ensemble is replaced by that of a ballroom in which each of the dancers performs his dance step according to his own pattern, interacts with the other participants (avoids collisions and finds its place), all without any manager coordinating the whole thing. In this vision of *distributed* rather than *centralized* control, interactions between all are

made as equals, resulting from actions and reactions in relation to the perception of the immediate environment of each, known information, and the logic specific to each element. To return to the image that gave the choreography its name, it would be very difficult to build such an effective system for the proper management of the dancers at the ball using a centralized algorithm. This would require a large number of communications and the processing of a lot of information for a general result that would probably be more random and less fluid.

Like the ball, the choreography in SOA is designed as a form of collaboration between the stakeholders without one of them prevailing over the others. Here too, each participant pursues his objective, reacts to the messages received, and uses the information transmitted by the partners. The difficulty lies in designing algorithms specific to each node, describing its task, while at the same time achieving the common goal (if there is one). The idea is that everyone respects their own logic of behavior, according to the information accessible from their own point of view. The sum of the valid behaviors allows the whole to remain within satisfactory bounds. Figure III.4 visualizes exchanges during a service choreography approach.

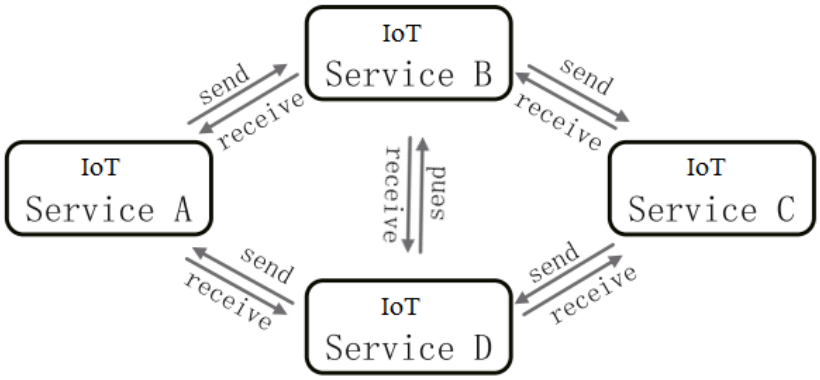

Figure III.4: IoT Service choreography.

In summary, we believe the choreographed approach is well suited for the Internet of Things. At the same time, it facilitates access to the resources of his WSAN network related to IoT [193]. With choreography, direct interaction between nodes is less taxing on the network than uploading data to a central point, thus meeting power constraints. In addition, the processing capabilities added to objects (certainly limited, but present) allow the implementation of algorithms that describe the tasks performed by the objects themselves [165].

III.3 IoT services composition: a panel of approaches

The problem of QoS-aware IoT service composition is widely discussed in the literature. Indeed, QoS-aware service composition is the process by which atomic services are combined to provide new functionality that none of the services can provide individually while still guaranteeing a better overall QoS. Much of the literature dedicated to the IoT services composition focuses on the development of formal composition methods and algorithms.

The goal of the approaches is to be able to automatically instantiate composition plans based on an abstract plan. The abstract plan is defined as a set of abstract tasks or services to be instantiated with real IoT services.

III.3.1 Literature review

In this section, we review the main existing works that tackle the QoS-aware IoT services composition under the QoS constrains.

Among the solutions proposed in the literature to raise the challenge of QoS-aware IoT services composition, bio-inspired meta-heuristic approaches: approaches based on Genetic Algorithms (GA) [194, 195, 196, 197]; GA and Particle Swarm Optimization (PSO) based [198, 199]; and Cuckoo-Inspired approaches [200].

To address user QoS constraints, the authors in [194] formulate the IoT Service Composition Oriented QoS (QSC) problem as a Multi-Criteria Goal Programming (MCGP) model, and the Multi-Population Genetic Algorithm (MGA) solves it.

The authors in[198] propose a collaboration of genetic algorithms (GA) and particle swarm optimization (PSO) to solve the problem of QoS-aware IoT service composition. According to the authors, the proposed approach takes into account: (1) Proportional evolutionary optimization to ensure population diversity; (2) a better and improved local strategy for candidate service selection, (3) an optimal disruptive global strategy along the optimal global particle, and (4) a self-adaptive learning rate adjustment mechanism that improves the convergence speed and the optimized result. All this happens when service composition is done at a large scale. On the other hand, [199] compared two algorithms, GA and PSO, in solving IoT service composition problems. The authors confirm that the two algorithms solve the problem. But, the experimental results using real datasets show that the solutions obtained with GA are more expressive than those obtained with PSO.

In [195], the authors use a Multi-Attribute Decision-Making method (MADM) to evaluate each IoT service and select elites in terms of QoS as part of the composition. As a global optimization method, a genetic algorithm (GA) improved by adding a greedy algorithm is adopted. The result obtained is the service composition that maximizes the value of the utility function.

The authors in [196] propose a QoS aggregation scheme to compute the total QoS vector of composite service. Composition optimality is determined using the QoS utility function, which takes into account user constraints and the weight vector. Finally, a genetic algorithm is developed to have the optimal service composition.

The authors of [197] propose a multi-objective meta-heuristic search algorithm to solve the problem of QoS-based composition of IoT services. The *Non-dominated Sorting Genetic Algorithm* (NSGA-II) is used for having the optimal composition.

In [200], the authors develop an algorithm inspired by cuckoo behavior to find existing composite services that efficiently satisfy user demands in a Multi-Cloud Environment (MCE). The proposed MultiCuckoo algorithm takes into account service composition issues and their details in a Multi-Cloud Environment (MCE). The evaluation indicates less

consumption of time and resources.

Other approaches try to reduce the composition space by removing unpromising candidate services in terms of QoS. For that, clustering techniques are used to group services according to their relevance in the composition process and only consider relevant clusters [8, 201, 202, 203].

The authors in [8] suggest *Clustering-based and QoS-aware service Composition Algorithm* (CQCA) in Ambient Intelligence environment (AmI). For this purpose, the K-means method is used to group the services according to their QoS levels i.e., each cluster represents a QoS level. Using the properties of the outcome clusters, a new utility function was proposed to eliminate services that were unpromising in terms of QoS. This technique further reduced the composition space and time. They then use a lexicographical method to verify compliance with global constraints. Finally, a search tree was constructed to determine sub-optimal composition.

The authors in [201] propose a fast and reliable service composition approach for integrating physical, cyberspace, and social networks. First, remove redundant components using the Skyline component computation. Then, use the coefficient of variation to filter out components with highly variable quality of service (QoS). Finally, the best component is chosen by maximizing the fitness function according to the user's end-to-end QoS requirements using *Mixed Integer Programming* (MIP).

In the context of large-scale IoT environments, the literature [202] suggests *Two-Steps QoS-aware services Composition Algorithm* (TS-QCA) considering global QoS constraints. The TS-QCA algorithm is based on the Shuffled Frog Leaping Algorithm (SFLA) and clustering method, which reduces the composition time by focusing only on promising candidate services in the composition process, thus improving the algorithm convergence.

Another challenge in IoT is optimal energy management as part of composing IoT services [203, 204, 205, 206, 207].

The literature [203] combines both clustering methods and energy management. The traditional QoS model is extended by the introduction of energy and a method for its calculation is proposed. Considering the new QoS model, global constraints are decomposed into local QoS constraints for atomic services. The K-Means method is used to get a clusters of atomic services. Finally, the best service is selected locally on each cluster.

In the paper [204], the authors propose Fast Energy-Centered and QoS-aware Service Composition Approach (FSCA-EQ) for IoT service composition. A hierarchical optimization mechanism is used. First, it pre-selects candidate services with higher QoS using a compromise ratio (CRM) method. To select the best service as the final composite service, the concept of relative dominance is applied depending on its energy profile, QoS attributes and user preferences. FSCA-EQ successfully balances the QoS level and power consumption of composite service in a large scale IoT environment.

In [205] the authors proposed *Energy-centric and QoS-aware IoT services Selection Algorithm* (EQSA) for composition of IoT services. Services whose quality of service satisfies users are preselected using lexicographical optimization strategies and QoS con-

straint relaxation techniques. The problem is believed to be a multi-objective optimization problem. Minimize power consumption to ensure high availability of composite services while meeting QoS requirements.

In [206], the problem of finding a balance between the QoS level and the energy consumption for IoT service composition is reduced to a shortest path optimization problem with two objectives. The authors use a precise algorithm called Pulse to solve the problem.

The authors in [207] have developed a multi-cloud IoT service composition algorithm called (E2C2). E2C2 is an energy-sensitive configuration layer that aims to find and interact with as few IoT services as possible. A formal translation of user needs, transformation modeling and analysis are adopted for the proposed algorithm. A minimal of IoT services that meet user demands are explored and composed in an energy efficient manner.

Other approaches use different visions to solve the composition of IoT service problem. As in [208], an approach to self-management of trustworthy systems is proposed using collaborative filtering and his goal-oriented IoT service composition. Collaborative filtering based on matrix factorization (MF) is used in a goal-oriented self-managing service model for performing tasks in IoT. Goal-oriented models dynamically adjust based on QoS predictions to enable disaster recovery, reduce outages, and reduce reruns.

The authors in [209] propose a crowdsourced IoT services framework to select and assemble crowdsourced mobile IoT services based on spatio-temporal factors. Algorithms based on deep reinforcement learning have been developed to select and assemble mobile services considering QoS parameters without using indices. To detect services in motion, Spazio-temporal MapReduce is also developed based on the swarm pattern using Apache Spark. The proposed approach is scalable and highly accurate compared to ground truth.

The authors of [210] decompose the complex QoS calculation model into four basic models and assigns a suitable method for each QoS calculation model according to IoT properties. They then use a heuristic backtracking (BT) algorithm to meet the QoS requirements.

In environmental sensor networks, the authors in [211] present a *Service Monitoring And Composition System* (SMACS). Reliability and Reactivity are the two parameters estimated by SMACS using a simple Bayesian classifier. These parameters are used to get the optimal service composition. This can improve the overall QoS of the entire sensor network by filtering out bad nodes.

III.3.2 Approaches studied, a comparison

The QoS-aware IoT service composition approaches presented in the literature review section (Section III.3.1) are summarized in Table III.2, and are compared in terms of: (1) resolution principle that represents the used solving method, (2) scalability, (3) composition optimality i.e. compare the composition obtained with respect to the optimal one.

Apprch	Methods used	Scalability	Optimality
[204]	CRM + Relative dominance	Yes	Optimal
[205]	Lexicographic optimization + relative dominance	Yes	Close-to-optimal
[8]	k-means + Utility function + Lexicographic optimization methods	Yes	Near-to-optimal
[201]	Skyline component + QoS fluctuation + MIP algorithm	Not evaluated	Not evaluated
[202]	Clustering + SFLA	Yes	Near-to-optimal
[194]	MCGP model + MGA	Yes	Optimal
[198]	GA + PSO (Cooperative evolution algorithm)	Not evaluated	Not evaluated
[199]	GA + PSO Algorithms	Not evaluated	Not evaluated
[195]	GA + Greedy algorithm	Not evaluated	Not evaluated
[196]	QoS aggregation schema and Utility Func-tion + GA	Yes	Optimal
[197]	Non-dominated Sorting GA NSGA-II	Not evaluated	Not evaluated
[200]	MultiCuckoo algorithm	Not evaluated	Near-to-optimal
[203]	Extended QoS model + K-means	Not evaluated	Optimal
[206]	Pulse Algorithm	Yes	Optimal
[207]	E2C2 Algorithm	Not evaluated	Optimal
[208]	Matrix Factorization	Not evaluated	Not evaluated
[209]	Deep Reinforcement Learning + Spatio-temporal MapReduce	Yes	Optimal
[210]	BackTracking Algorithm	Yes	Sub-optimal
[211]	Naive Bayesian Classifier	Not evaluated	Not evaluated

Table III.2: Approaches summarize and comparison

Discussion

To reduce the search space of the IoT service composition problem, several approaches have been proposed. The IoT service composition approach uses clustering techniques to eliminate candidate services or compositions with unpromising QoS values, thereby reducing the search space and consequently composition time, such as in : [8], [201], [202], and [203]. The k-means method is the most commonly used clustering method and had to be rerun every time clustering was performed. On the other hand, during the composition process, bio-inspired metaheuristic-based approaches can introduce inappropriate services into the process, like in : [194, 198, 199], [195, 196, 197, 200].

Moreover, most of the above approaches are limited because they specify the QoS attributes to be studied each time (runtime, cost, availability, etc.). So there is no a general and flexible approach among them that can be applied to any QoS models.

III.4 Conclusion

IoT makes computing accessible anywhere and anytime. This gives the possibility to the user to be located in different places while always guaranteeing his access to information, providing him with services that meet his preferences in terms of QoS. In this chapter, we first presented the fundamental concepts of the IoT service, and the problem of the IoT services selection and composition. The second part, was devoted to study some of IoT services selection and composition approaches existing in the literature. A comparison between the different approaches studied is carried out according to the resolution methods used, the scaling, the optimality and the QoS properties considered. According to the resolution methods used, we can classify the approaches studied in four categories: (1) approaches based on bio-inspired metaheuristics; (2) approaches based on a combinatorial model or graphs; (3) approaches based on dominance in the Pareto; and (4) approaches based on planning techniques derived from Artificial Intelligence. Artificial intelligence techniques that support the dynamic composition of large-scale IoT services. In this category of approaches lies our proposal, the details of which will be the subject of the next chapter.

Chapter IV

GNN-QSC: QoS-aware IoT Services Composition Approach

IV.1 Introduction

The problem of service composition has been widely addressed in the context of IoT in recent years. Indeed, this problem amounts to selecting for each request, a composite service which best satisfies the QoS requirements imposed by the user. In this chapter, we propose a new QoS-aware IoT service composition approach (*GNN-QSC - Genetic Algorithms and Neural Networks for QoS-aware IoT Services Composition*). This approach will be based on AI techniques, of which we will use the neurons networks as a clustering technique in order to reduce the composition space and thus reduce the composition time.

In this chapter, first, we present the service composition problem taking into account global QoS constraints. Next, we give an overview of the techniques used (genetic algorithm and neural networks). Finally, we detail the phases of the proposed approach GNN-QSC.

IV.2 Service composition problem modeling

Service composition is defined as a process of combining atomic services to create a new composite service that provides new functionality that no atomic service can provide individually. QoS-aware service composition consists in selecting a composite service in such a way as to satisfy the functional and non-functional requirements of the user. Functional requirements are represented as an abstract composition, while non-functional requirements are formulated as user-imposed constraints on QoS attribute values.

A dynamic model of IoT Service composition is shown in the Figure IV.1. The same shape indicates the same functionality and different colors indicate differences regarding QoS. So a user queries a directory containing N service classes (tasks T_i) to select m tasks to be composed ($m \leq N$) in a certain order (composition structure) that meets the needs of your business process. Each task T_i ($i \leq m$) is represented by a class of services called *Abstract Service* (AS). Each class contains n functionally identical IoT services, but with different non-functional properties (QoS). Dynamic composition of IoT services

consists of choosing the best service in the corresponding Abstract Service (AS) at runtime. Therefore, the composite service result satisfies the global QoS constraints and provides the best overall QoS.

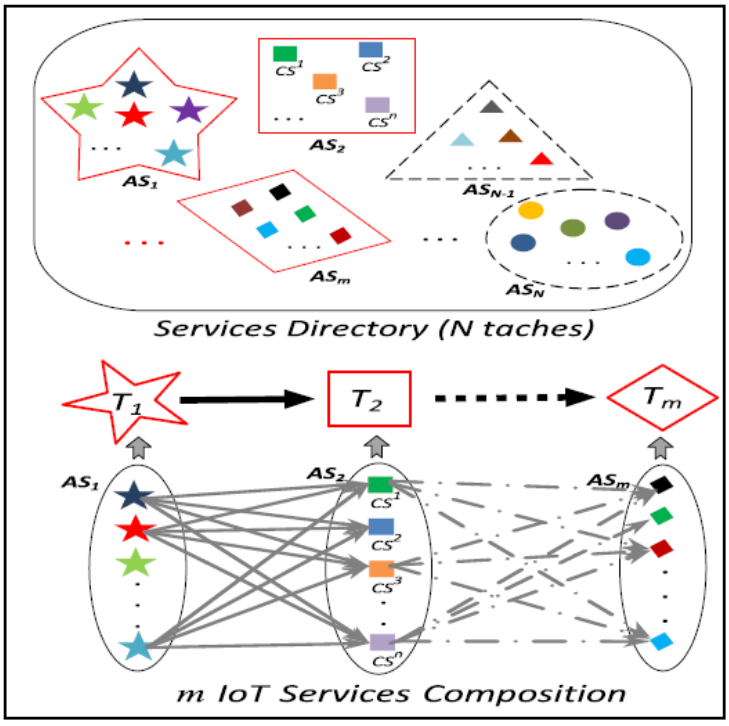

Figure IV.1: Dynamic IoT services composition process.

IV.2.1 Concrete and Abstract Service

- A ***Concrete Service*** **(CS)** is an instance of an Abstract Service. It is represented by a tuple *CS= [F, I, O, QoS]* , where: *I/O* is respectively the vectors of input and output parameters of a CS. F is the function, which will transform the inputs I into outputs O, in other words, F represents the functional properties of the service CS. Finally, QoS is the vector of quality of service attributes (non-functional properties of a concrete service).

- An ***Abstract Service*** **(AS)** is a set of n similar concrete services (CS) in term of their functionality, $AS_i = \{CS_i^1, CS_i^2, CS_i^3, ..., CS_i^n\}$. Thus, the $CS_i^{k\in[1..n]}$ have the same functionality (F), the same inputs and outputs (the vectors I, O), but their QoS attributes (vector QoS) are different.

IV.2.2 Abstract Composite and Concrete Composite Service

- ***Abstract Composite Services*** **(ACS)** is a set of m abstract services, $ACS =< AS_1, AS_2, ..., AS_m >$. They are connected, to each other, by one of the service composition structures: sequential, parallel, conditional, loop or a combination of these structures. The ACS is formulated according to the user functional requirements by using a set of connected abstract services.

- ***ccs*** is the instantiation of the Abstract Composite Service. The instantiation is done by the invocation for each abstract service AS_i the appropriate concrete service CS_i^k ($i \in [1..m], k \in [1..n]$), CCS = $<CS_1^k, ..., CS_i^k, ..., CS_m^k>$.

IV.2.3 QoS Attributes Vectors

- **QoS Attributes Vector of a Concrete Service**, noted *CS[QoS]*. It is defined by the QoS attributes vector. Formally, $CS_i^k[QoS] = [qos_{i1}^k, ..., qos_{ip}^k, ..., qos_{iq}^k]$, where $i \in [1..N], k \in [1..n], p \in [1..q]$, N is the abstract services number in the directory, n is the concrete services number in the abstract service AS_i and q is the QoS attributes number. The qos_{ip}^k is the value of the p^{th} QoS attribute of the k^{th} concrete service that belongs to the i^{th} abstract service. We distinguish between two types of QoS attributes: (1) *negative attributes* (cost, response time, etc.) which have a negative effect on the level of the QoS, i.e., more their values are higher more the QoS is lower. (2) *Positive attributes*, such as availability, reliability, etc., which have a positive effect on the level of the QoS, i.e., more their values are higher more the QoS is better. For this, we need to assign to each concrete service a utility function *fu*.

- **QoS Attributes Vector of a Concrete Composite Service**, is defined as follows: CCS.QoS =$[Q_1, ..., Q_q]$, where q is the QoS attributes number and Q_p is the value of the p^{th} attribute after aggregation. The aggregation of attributes depends on the used function (Summation, Product, Maximum or Minimum) and on the composition structure (Sequential, Parallel, Conditional and Loop). Table IV.1 summarizes the different aggregation functions of m services. P_p is the probability to select a concrete service in the conditional composition structure. And, *Nbr* is the number of iterations in the loop structure [8]. However, parallel conditional and loop structures can be transformed into a sequential model as presented in [212]. In the same way, a utility function *FU* is associated with each Concrete Composite Service calculated from its aggregated QoS values [8].

	Composition structure			
Function	**Sequential**	**Parallel**	**Conditional**	**Loop**
Summation	$\sum_{p=1}^{m} qos_{ip}^k$	$\sum_{p=1}^{m} qos_{ip}^k$	$\sum_{p=1}^{m} P_p * qos_{ip}^k$	$Nbr * qos_{ip}^k$
Product	$\prod_{i=1}^{m} qos_{ip}^k$	$\prod_{i=1}^{m} qos_{ip}^k$	$\sum_{p=1}^{m} P_p * qos_{ip}^k$	$(qos_{ip}^k)^{Nbr}$
Maximum	$\sum_{p=1}^{m} qos_{ip}^k$	$\max qos_{ip}^k$	$\sum_{p=1}^{m} P_p * qos_{ip}^k$	$Nbr * qos_{ip}^k$
Minimum	$\min qos_{ip}^k$	$\min qos_{ip}^k$	$\min qos_{ip}^k$	qos_{ip}^k

Table IV.1: Qos-Aggregation Functions.

IV.2.4 Utility Functions

- **Utility Function of a Concrete Service**, in order to evaluate a concrete service CS_i^k by using its *QoS* vector, a utility function *fu* is associated to it. The role of

this function is to transform all the values of its *QoS* vector into a single value. This latter allows the comparison between services that belong to the same Abstract Service (AS) in term of QoS. In this thesis, we use the *Simple Additive Weighting* (SAW) technique . It is widely adopted, in the literature, in the field of service composition [180]. The utility calculation involves:

1. The normalization of the QoS (*Nqos*) attributes values in values between 0 and 1, by comparing each value of QoS with the maximum and minimum values of this attribute. This allows making them independent from the units and ranges of measurements. However, as mentioned above, positive attributes should be maximized, while negative attributes should be minimized. Thus, the normalization will be done in two different ways, as defined in IV.1.

$$Nqos_{ip}^{k} = \begin{cases} 1, \ if(qos_p^{max} - qos_p^{min}) = 0 \\ Else \begin{cases} \frac{qos_p^{max} - qos_{ip}^{k}}{qos_p^{max} - qos_p^{min}} & \text{if } p \text{ is a negative attribute.} \\ \frac{qos_{ip}^{k} - qos_p^{min}}{qos_p^{max} - qos_p^{min}} & \text{if } p \text{ is a positive attribute.} \end{cases} \end{cases} \quad \text{(IV.1)}$$

   Where qos_{ip}^{k} represents the current value of the p^{th} QoS attribute of the k^{th} concrete service CS_i^k in the abstract service AS_i, qos_p^{max} and qos_p^{min} represent the maximum and minimum values respectively of the p^{th} QoS attribute.

2. The weighting to represent priorities and preferences of the user w_p, (such as: $\sum_{p=1}^{q} w_p = 1$, p = 1..q) are weights assigned to each QoS attribute. The differences in weights represent the user preferences.

3. Finally, the utility of the concrete service is obtained by the weighted sum of the normalized QoS values, as given in formula IV.2:

$$fu(CS_i^k) = \sum_{p=1}^{q} w_p * Nqos_{ip}^{k} \quad \text{(IV.2)}$$

- **Utility Function of a Concrete Composite Service**, the utility of a given Concrete Composite Service CCS= $< CS_1^k, ..., CS_i^k, ..., CS_m^k >$ is calculated, as in the utility function of a concrete service, by mapping the values of the *QoS* vector aggregated $[Q_1, ..., Q_p, ..., Q_q]$ into a single value by using the same technique SAW. Sun and Zhao in [213] have proved that the global utility of a composite service is the sum of the local utilities of the concrete services forming the composition. It is given by formula IV.3:

$$FU(CCS) = \sum_{i=1}^{m} fu(CS_i^k) \quad \text{(IV.3)}$$

IV.2.5 Global QoS Constraints

The user's QoS preferences are formulated as global constraints. These latter are imposed on the values of the QoS attributes of the composition (after aggregation). So, the Global QoS constraints is: CG (QoS) = $[Cg_1, ..., Cg_q]$; where q is the attributes number, and Cg_p is the global constraint imposed by the user on the p^{th} attribute of the composite service Q_p. It can be a lower bound ($Q_p \geq Cg_p$), if the attribute is a positive attribute, or an upper bound ($Q_p \leq Cg_p$) in the case where the attribute is negative.

IV.2.6 Feasible, Near-Optimal Concrete Composite Service

- **Feasible Concrete Composite Service**, We consider that a concrete composition (CCS) is feasible if for each abstract service ($AS_i, i = 1..m$), in the composition, a single concrete service (CS_i^k) is chosen. In addition, the aggregated QoS values of the composition (Q_p) satisfy the overall QoS constraints (Cg_p) imposed by the user (i.e. $Q_p \geq Cg_p$ if the attribute is positive,$Q_p \leq Cg_p$ otherwise) [8].
- **Concrete Composite Service (CCS) is called a near-to-optimal** one if it is feasible and has best overall utility (max (FU)) [8].

IV.3 GNN-QSC Approach Overview

Depending on the the composite service creation time, we can distinguish between two types of service composition: *static* composition and *dynamic* one. In static composition, services aggregation is done at design time. This compositing mode is suitable when the number of tasks to be composed to satisfy the user's requirements is unchanged and fixed. If changes are expected at runtime static mode is not adequate if it is necessary to replace unavailable services or has better alternatives. Dynamic configuration, on the other hand, allows selection and aggregation of services during the execution process. This is done automatically using semantic web technologies and artificial intelligence techniques. Dynamic composition is the ideal mode for composing IoT services. This is due to the natural dynamics of the IoT environment. However, dynamic composition of IoT services presents some challenges, how establish an ideal configuration that meets the user's QoS settings? To meet this challenge, we propose a Genetic algorithm and Neural Network-based approach for QoS-aware IoT Service Composition (GNN-QSC). This provides a near-optimal composition of IoT services, taking into account the QoS constraints imposed by user demand.

The IoT service composition process selects appropriate concrete services CS_i^k from each abstract service AS_i (i = 1..m) belonging to the composition plan (Abstract Composite Service ACS) to maximize the overall QoS gain of the Concrete Composite Service CCS, while respecting the QoS preferences imposed by the user (QoS global constraints) ($Q_p \leq Cg_p$ if the qos_p attribute is a negative attribute that one wishes to minimize, $Q_p \geq Cg_p$ if the qos_p attribute is a positive attribute to be maximized).

First, we develop a neural network model that allows us to rank each IoT service based on its QoS attribute vector. Then assign it to a quality of service level category. The QoS

attributes of IoT services are similar to the QoS attributes of web services. However, they differ when it comes to QoS attributes related to smart objects such as power consumption and location [214]. Therefore, the QWS datasets[1] can be used [215]. The QWS Dataset 1.0 is used for supervised training of the Neural Network model. Meanwhile, QWS Dataset 2.0 is used to evaluate the effectiveness of the approach. For training of the neural network, the QWS dataset 1.0 contains services decreed by the QoS attribute vectors (input vector) and their ratings (value labels: bronze, silver, platinum, or gold).

Second, for each attribute we decompose the interval $[qos_p^{min}, qos_p^{max}]$ into M QoS-levels (where qos_p^{min}, qos_p^{max} are, respectively, the minimum and the maximum value of the p^{th} attribute, p = 1..q and q is the attributes number).

Third, we use the theoretical M service generated in the previous step (the new QoS vector after decomposing it into M levels) and apply a genetic algorithm to create *Ideal Theoretical Composition* (ITC). The theoretical services that make up the ITC are evaluated using neural network model, in order to deduce the categories of the real concrete services for each abstract service to be engaged in the concrete composition.

Finally, we use the genetic algorithm again to get the optimal composition we need. The Figure IV.2 outlines the proposed approach.

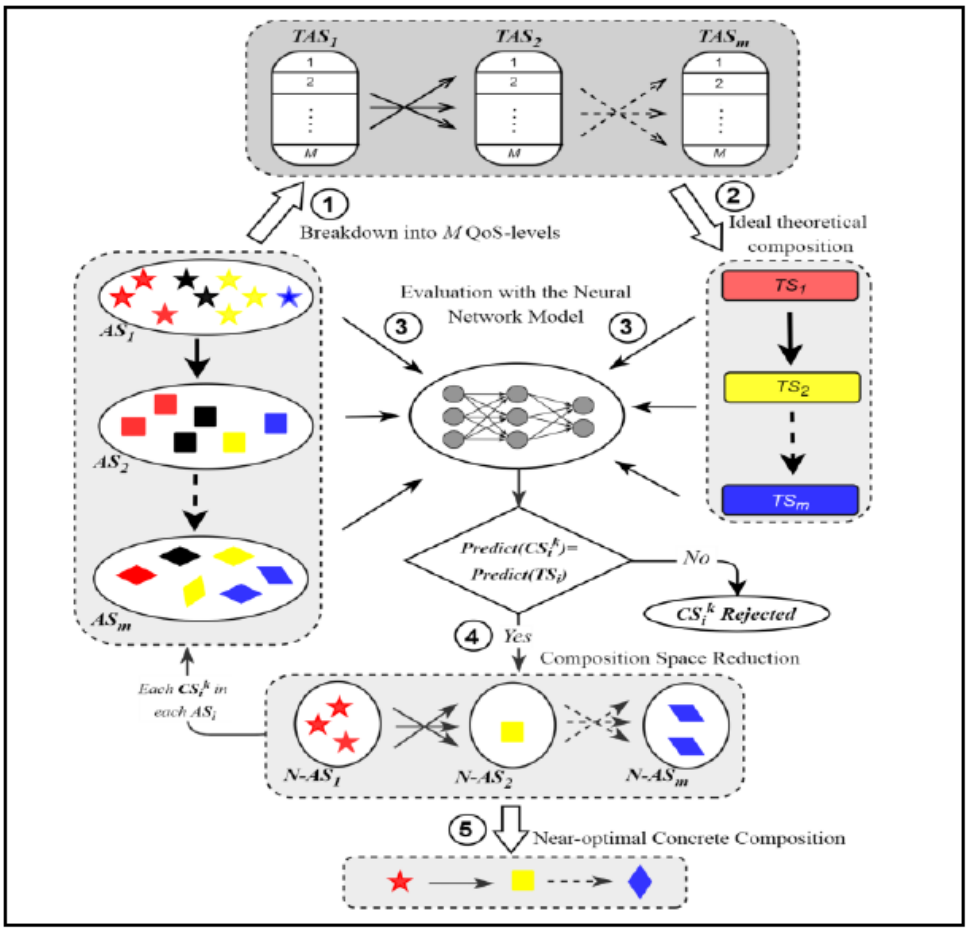

Figure IV.2: GNN-QSC overview.

[1]The QWS Dataset, https://qwsdata.github.io/, Last update: 01 November 2019.

Clustering techniques allow concrete services (CS) (services with the same functionality) belonging to an abstract service (AS) to be grouped according to QoS parameters within a set of clusters called QoS levels. Each level groups services with similar QoS attribute values. We propose a neural network model that ranks IoT services and groups them according to their corresponding QoS levels. We chose neural networks as classifiers because they can correctly predict new situations based on the knowledge acquired during the learning process [215]. In addition, the solution parameters are iteratively generated.

The neural model contains one hidden layer of neurons which use the sigmoidal activation function. The number of neurons in the input layer corresponds to the vector dimension of QoS attributes (q neurons). The output layer contains the number of required classes. This corresponds to the number of classes in the dataset used in the neural network's supervised learning process. As previously mentioned, we use the QWS dataset 1.0, which represents measurements of nine QoS attributes of 365 real-world services. They are categorized into four classes: Platinum, Gold, Silver, and Bronze. The neural model should be able to place a particular service into one of these classes. Platinum Class includes higher levels of quality of service (faster response times, lower prices, higher availability, etc.). However, Bronze includes poor quality service (long response time, high price, low availability. . .). The figure IV.3 shows an overview of a neural network.

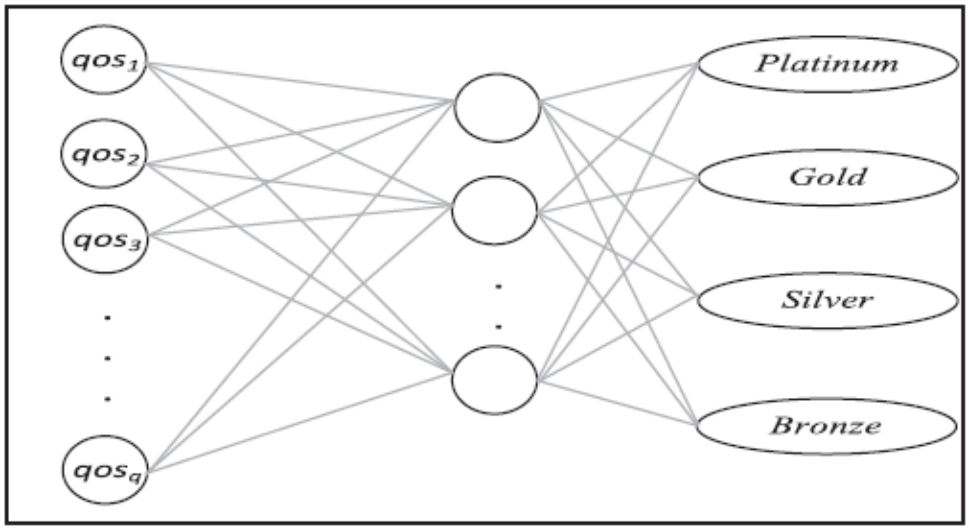

Figure IV.3: Neural network model.

IV.3.1 QoS-Intervals breakdown into M levels

In this step, we decompose for each Abstract Service (AS_i), and for each QoS parameter ($qos_{ip}, p = 1..q$) the interval $[qos_{ip}^{min}, qos_{ip}^{max}]$ into M QoS levels using formula IV.4. Thus, we can form *M-1* sub-intervals of QoS for each attribute; qos_{ip}^{max} and qos_{ip}^{min} are, respectively, the greater and the smallest values of the p^{th} attribute in the QoS vectors of all the concrete services belonging to the abstract service AS_i. M is a constant number that is fixed a priori. In order to ensure a balanced distribution of real values of qos_{ip} in the formed sub-intervals, we choose M according to the distribution degree of qos_{ip} values in the interval $[qos_{ip}^{min}, qos_{ip}^{max}]$.

$$qosth_{ip}^{l} = qos_{ip}^{min} + \frac{l-1}{M-1} * (qos_{ip}^{max} - qos_{ip}^{min}) \quad \text{(IV.4)}$$

$$1 \leq l \leq M, 1 \leq p \leq q, 1 \leq i \leq m$$

By doing the same operation for all the attributes (q attributes) and for each abstract service in the composition plane (m abstract services). We will have M new vectors of the theoretical QoS values ($qosth_{ip}^{l}$). These vectors are included between the values of qos_{ip}^{min} and qos_{ip}^{max} of each abstract service, which we consider as the M Theoretical Services TS_i^l ($l = 1..M$) of the Theoretical Abstract Service TAS_i.

IV.3.2 Find the ideal theoretical composition

At the end of the previous step, we get m theoretical abstract services (TAS_i, $i = 1..m$) that containing theoretical services (TS_i^k, $k = 1..m$). These theoretical services are obtained from concrete services belonging to abstract services (AS_i, $i = 1..m$) of the composition plane. The Genetic Algorithm (GA)-based heuristic approach is applied to obtain the Ideal Theoretical Composite (ITC) service with the best theoretical QoS performance.

In GA, the solution should be represented as individuals, and the fitness function evaluates the relevance of each individual. After creating the initial population, targeted genetic operators (selection, crossover, mutation) are applied to develop compositions and find the best one.

IV.3.2.1 Individuals coding

Individuals (chromosomes) represent feasible composition, that's to say, composition that satisfies the general constraints. An individual is formed by concatenating the theoretical QoS vectors (*qosth*) of the Theoretical Services (*TS*) of each Theoretical Abstract Service (*TAS*), according to the composition plan, as shown in Figure IV.4.

In the GNN-QSC approach, each individual consists of $m * q$ genes (where m is the number of tasks to composed and q is the QoS attributes number). Each gene contains a theoretical QoS value resulting from a decomposition step into M QoS-levels.

IV.3.2.2 Initial Population

The initial population is formed by a set of valid compositions (individuals) selected in a random fashion. That is, from each theoretical abstract service (TAS) select a theoretical candidate service (TS) for composition. Then use the aggregation formula shown in table IV.1 to calculate the aggregated QoS values for this composition and check if they satisfy the general QoS constraints. Otherwise, that individual is eliminated and another is built. Repeat this process until we have the desired number of valid individuals that make up the initial population.

IV.3.2.3 Fitness Function

Since an individual represents a valid service composition, the fitness function must first ensure that the global QoS constraints are met after computing the aggregate QoS value.

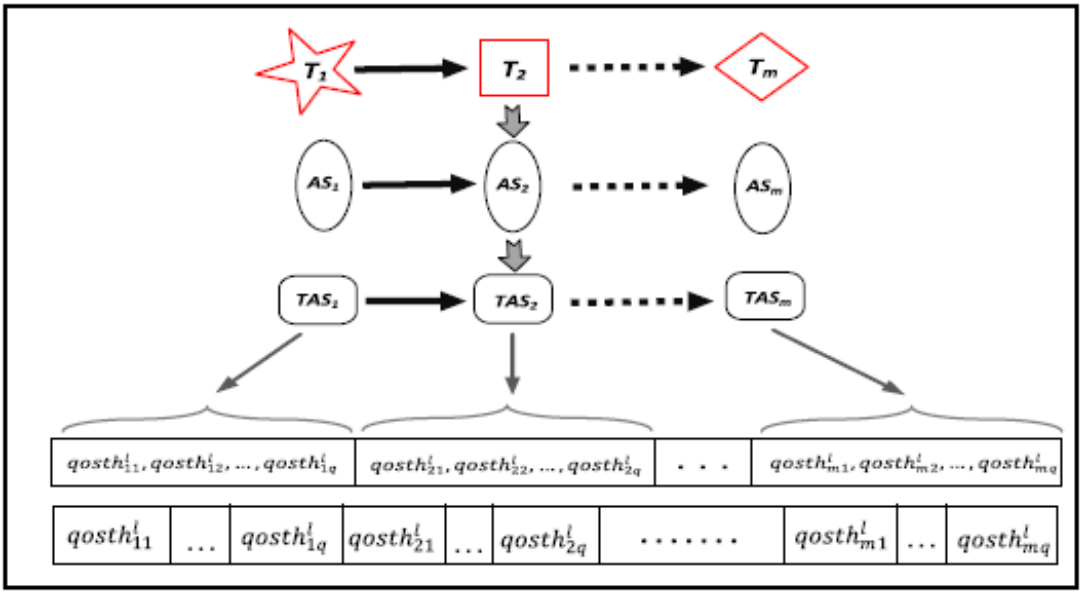

Figure IV.4: Individuals Coding.

We then compute the chromosome fitness using the utility function FU associated with the composite service of the formula IV.3.

IV.3.2.4 Genetic Operators

Our approach selects the two individuals with the highest fitness function values (elitist method) and choose them as parents for offspring production. This happens in every generation. Applying the multipoint crossover method to exchange information between the two parents for producing the two offspring. The crossing points are randomly chosen for each theoretical service in the composition. The new descendants (offspring) are then generated by pairwise exchanging the genes of the segments between these points.

For the mutation operator, we try to generate better individuals in each generation by maximizing the fitness function. This can be achieved by optimizing both positive and negative attributes. To do this, three genes (QoS attributes) are randomly selected for each service in the composition (individual). If the selected gene is a positive attribute, mutation is performed with randomly generated value between the maximum and current value of this attribute for the abstract service that the associated service belonging to. Otherwise, for negative attributes, the mutation is performed using a randomly generated value between the minimum and current value for that attribute.

IV.3.3 Evaluation with the neural model

After the genetic algorithm convergence, we have theoretical composition, of m theoretical services, that satisfies the general QoS constraints and have near-optimal quality of service. It is the Ideal Theoretical Composition (ITC). Let $ITC =< TS_1, TS_2, .., TS_m >$ is the ideal theoretical composition found. Our approach evaluates each theoretical service of the ITC TS_i $(i = 1..m)$ by a neural network model *(Predict* (TS_i)*)*. As a result you will know the category (Platinum, Gold, Silver, Bronze) of each one. Similarly, the same neural

network can be used to find concrete service categories for abstract services.

IV.3.4 Composition space reduction

The QoS vectors that form the ITC are theoretical (*qosth*) which will be used to infer concrete services categories for each abstract service included in the optimal concrete composition. In this way we eliminate bad services, the thing that will shrink the workspace. Each abstract service holds only concrete services that have the same category as the corresponding theoretical service. This reduces the search space for each abstract service by three quarters. (See Algorithm.1).

Algorithm 1

Inputs: $ACS = <AS_1, AS_2, \ldots, AS_m>$

// Abstract Composite Service (m Tasks)

$AS_i = \{ CS_i^1, CS_i^2, \ldots, CS_i^{n_i} \}$

// Abstract Service i (contains n_i concrete services)

$ThC_{ideal} = <TS_1, TS_2, \ldots, TS_m>$

// Ideal theoretical composition

1: **for** $i = 1$ **to** m
2: $N\text{-}AS_i \longleftarrow \emptyset$
3: **for** $k = 1$ **to** n_i
4: **if** $Predict(CS_i^k) = Predict(TS_i)$ **then**
5: $N\text{-}AS_i \longleftarrow N\text{-}AS_i \cup CS_i^k$
6: **end if**
7: **end for**
8: **end for**

Outputs: $N\text{-}ACS = <N\text{-}AS_1, N\text{-}AS_2, \ldots, N\text{-}AS_m>$

Figure IV.5: Algorithm 1.

IV.3.5 Search the Near-Optimal concrete composition

To get a near-optimal composition, we use the genetic algorithm twice. This allows population reduction (result obtained in the previous step). Individuals in the initial population are randomly formed by chaining concrete services of a new abstract service ($N - AS_i$, $i = 1..m$). The individuals are feasible compositions. We maintain the same genetic algorithm policy that was previously applied regarding the application of genetic operators. As the genetic algorithm converges, the result is a composition of concrete service candidates that satisfy the general QoS constraints with near-optimal utility.

IV.4 Conclusion

IoT service composition is the combination of atomic IoT services to create a new service that meets a specific need. Often, the user has preferences in terms of QoS, he expresses them under QoS-global constraints on the composite service. However, solving service composition under global constraint is known as an NP-hard problem. In this chapter we have presented some concepts related to the QoS-aware IoT services composition in order to be able to model our proposed approach based on neural network and genetic algorithm. A general description is given. In the following chapter, an illustrative scenario and a possible evaluation will be made.

Chapter V

Experimental results and performance analysis

V.1 Introduction

In the previous chapter, we presented the general design of the proposed IoT service composition approach (GNN-QSC). This chapter is dedicated to the simulation and validation of the GNN-QSC service composition approach proposed. It is composed of two parts. The first part is devoted to an illustrative scenario to show the functionality steps of the proposed approach. However, the second is to evaluate the performance of the proposed approach through a set of experiments, in which we will pass our approach to the comparison with other existing IoT service composition approaches in the literature.

V.2 Illustrative scenario

To illustrate the steps to be followed according to our approach (GNN-QSC), we propose the following scenario: A user's query (ACS) has two tasks (AS), $ACS =< T_1, T_2 >$. Each task has nine concrete IoT services with different QoS vectors. These vectors are randomly selected from the QWS dataset 2.0 [215]. QWS Dataset 2.0 contains a set of 2507 services. Each row in this record represents a real service with nine corresponding QoS measurements. Assumed services are listed in the table V.1.

The global QoS constraints are supposedly the following:

CG(QoS) = [$ResponseTime \leq 9790.17ms$,

$Availability \geq 2\%$,

$Throughput \geq 5.72invokes/second$,

$Successability \geq 4.42\%$,

$Reliability \leq 44.32\%$,

$Compliance \geq 30.47\%$,

$BestPractices \geq 26\%$,

$Latency \leq 6233.67ms$,

AS	CS	QoS Attributs								
		RTime	Avail	Thrghput	Succ	Relia	Compl	BPract	Lat	Docum
T_1	T_{11}	107.57	80	1.7	81	67	78	82	18.21	61
	T_{12}	269.83	85	4.5	86	53	89	66	74.96	6
	T_{13}	42.5	72	13.5	72	73	78	84	2.5	11
	T_{14}	106.75	90	16.2	96	73	78	80	24.13	93
	T_{15}	408.21	56	5.0	58	73	78	80	121.46	89
	T_{16}	103.25	93	20.0	98	73	78	84	1.0	86
	T_{17}	114.33	85	16.2	95	73	100	84	13.0	39
	T_{18}	40.0	72	19.8	72	73	78	84	3.5	5
	T_{19}	672.2	91	7.9	97	73	100	84	9.2	36
T_2	T_{21}	408.0	83	15.2	84	83	89	91	3.0	6
	T_{22}	115.0	83	22.3	84	83	89	91	4.0	5
	T_{23}	46.05	89	20.7	96	67	100	77	4.48	8
	T_{24}	499.0	83	19.7	84	83	89	91	31.0	5
	T_{25}	3343.0	83	5.5	84	83	89	91	2.0	3
	T_{26}	109.6	99	19.3	100	73	78	62	1.0	94
	T_{27}	42.15	56	10.4	56	60	89	69	1.32	37
	T_{28}	102.0	90	18.6	97	73	78	80	1.0	92
	T_{29}	38.0	71	21.1	72	73	78	84	2.0	2

Table V.1: IoT Services descriptions.

Documentation $\geq 0.04\%$].

Response Time, Reliability and Latency are negative attributes. The remaining six are positives. For simplicity, consider all nine QoS attributes to have the same weight for user: ($w_p = 1/9$, $p = 1..9$).

The problem to solve is choosing the right concrete service for each abstract service. Therefore, you get a composite service with maximum QoS utility that respects the imposed global QoS-constraints.

To achieve our goal, the proposed approach (GNN-QSC) is to first build and train a neural model using the TensorFlow 2.3.0 library. A neural model allows each IoT service to be evaluated based on its QoS attribute vector. Then assign it to a QoS level category. Our neural model consists of a single hidden layer containing 32 neurons, which use a sigmoidal activation function. The first layer contains 9 neurons corresponding to the 9 QoS attributes we examined in the QWS 1.0 training dataset. The output layer contains four neurons corresponding to service classes (Platinum, Gold, Silver, and Bronze) of

the same dataset. A softmax activation function is used for the output layer neurons. Supervised training is run on 80% of the QWS 1.0 dataset and the 20% are used for testing. The model reported an accuracy of 91.33%. Note that the training phase runs only once.

The second phase begins as soon as the first one is reached. It consists of five steps that are as follows:

V.2.1 QoS-Intervals breakdown into *M* QoS-Levels

This step can reduce the search space, especially in large scale environments. Consider M to be an a priori constant such as $M \ll n$. In our scenario $n = 9$, we put $M = 4$. Then use the expression IV.4 to get four theoretical services for each abstract service, as shown in the table V.2.

TAS	TS	qosth values								
		ReTTh	AvlTh	ThrTh	SucTh	RelTh	CompTh	BPrTh	LatTh	DocTh
T_1	TS_{11}	40	56	1,7	58	53	78	66	1	5
	TS_{12}	250,73	68,33	7,8	71,33	59,66	85,33	72	41,15	34,33
	TS_{13}	461,46	80,66	13,9	84,66	66,33	92,66	78	81,30	63,66
	TS_{14}	672,2	93	20	98	73	100	84	121,46	93
T_2	TS_{21}	38	56	5,5	56	60	78	62	1	2
	TS_{22}	1139,66	70,33	11,1	70,66	67,66	85,33	71,66	11	32,66
	TS_{23}	2241,33	84,66	16,7	85,33	75,33	92,66	81,33	21	63,33
	TS_{24}	3343	99	22,3	100	83	100	91	31	94

Table V.2: QoS-Intervals breakdown.

V.2.2 Find the ideal theoretical composition

By using the Genetic Algorithm, we look for the ideal theoretical composition from the theoretical services generated in the previous phase (Table V.2). The individuals of the initial population are formed by concatenating randomly the theoretical QoS vectors (qosth) of the theoretical services (TS), of each theoretical abstract service (TAS), two by two ($< TS_{12}, TS_{24} >, < TS_{13}, TS_{21} > ...$). At this step, we check at each composition the satisfaction of the global QoS constraints. We use formulas of Table IV.1 to make the appropriate QoS attributes aggregations for the composite service. The evaluation of individuals (theoretical compositions) will be done by the composite service utility function (FU) by using formula IV.3 via formulas IV.1 and IV.2.

After GA convergence, we get a composite theoretical service that meets global QoS constraints with better utility. It is the Ideal Theoretical Composition (ITC). In this scenario, the ITC is displayed in the table V.3.

TS	qosth values								
	ReTTh	AvlTh	ThrTh	SucTh	RelTh	CompTh	BPrTh	LatTh	DocTh
TS_1	109.23	90.67	15.33	73.77	67.12	75.98	82.45	0.73	91.03
TS_2	50.59	60.76	9.34	71.33	64.29	85.12	69.59	2.78	39.54

Table V.3: Ideal Theoretical Composition (ITC).

V.2.3 Evaluation with the neural model

This step uses a neural network to evaluate the Ideal Theatrical Composition as well as concrete services by using the neural network. The results are shown in the table V.4.

AS	CS	Evaluation	ITC Services	Evaluation
T_1	T_{11}	Gold	TS_1	Gold
	T_{12}	Silver		
	T_{13}	Platinum		
	T_{14}	Gold		
	T_{15}	Silver		
	T_{16}	Gold		
	T_{17}	Gold		
	T_{18}	Platinum		
	T_{19}	Silver		
T_2	T_{21}	Silver	TS_2	Platinum
	T_{22}	Gold		
	T_{23}	Platinum		
	T_{24}	Silver		
	T_{25}	Bronze		
	T_{26}	Gold		
	T_{27}	Platinum		
	T_{28}	Gold		
	T_{29}	Platinum		

Table V.4: Services evaluation with the neural model.

V.2.4 Composition space reduction

This step reduces the concert composition space by keeping only the gold category concrete services from the first abstract service ($T_{11}, T_{14}, T_{16}, T_{17}$), and Platinum services from the second abstract service (T_{23}, T_{27}, T_{29}) according to ITC atomic services categories.

V.2.5 Find Near-optimal concrete composition

After selecting concrete candidate services for each abstract service, the genetic algorithm is again used to find a near-optimal composition . Individuals (compositions) are randomly formed from previously selected services. GA convergence results in suboptimal concrete composite service. Composition with greater QoS benefits by meeting the imposed constraints. The results is: $<T_{16}, T_{27}>$ with UF-utility = 1,098.

V.3 Experimental results and evaluation

This step in our work is to validate the effectiveness of dynamic IoT service composition under the global QoS constraint approach (GNN-QSC) in large scale environments.

The experimental context is designed as follows. Assume that you have an IoT service composition process. It consists of m abstract services (tasks). Every abstract service has n concrete services. To create these abstract services, we randomly select n services from the QWS dataset 2.0 each time. We kept the nine global QoS constraints used in the example scenario to make the evaluation more reliable (Section V.2).

As already mentioned, the genetic algorithm (GA) is used twice. The first goal is to obtain an ideal theoretical composition, from which to derive concrete service categories for each abstract service. A second GA is performed on the retained category of concrete services to obtain a near-optimal concrete composition. The configuration of GA is as follows:

V.3.1 GA Parameters setting

In general, key parameters of GA should be determined through experimentation. Existing research proves that the optimal number of individuals is between 20 and 160. The best crossover probability values are between 0.25 and 1.0, and mutation probability values between 0.005 and 0.1 [216].

Set the population size to 20, the crossover probability to 1.0, and the mutation parameter to 0.05. As a stopping criterion, we chose to stop the algorithm when the population can no longer evolve, i.e. If the difference in the utility of the solutions for generations g and g + 1 is very small (the stopping criterion chosen is when FU (g) - FU (g + 1) <= 0.0001).

Generating initial population:

For the first GA, the initial population is defined by the decomposition of each abstract service into M QoS-levels by applying formula IV.4. After obtaining the theoretical abstract services, we can produce the chromosomes (valid service compositions). We use a two-dimensional array to store a chromosome. In the first row, we store the valid theoretical composition. This latter is randomly generated from the theoretical services in each theoretical abstract service, according to the composition plan. The generated composition satisfies the overall QoS constraints. In the second row, we store the value of its Utility Function that is calculated through formula IV.3. Thereby, we repeat the

chromosome generation method until we obtain 20 individuals.

Before using the second GA, we first need to classify all concrete services for each abstract service using the proposed neural model. Then, for each abstract service, we retain the services having the same categories of the atomic theoretical services that make up the ITC. The latter was found in the previous step. The retained services are used to construct the individuals of the initial population for the second GA. Each chromosome is a composite of previously retained services. These latter are randomly generated so that the composition meets global QoS constraints. This process is repeated until 20 individuals are reached.

Selection Operation:

Carry over elitist selection as the selection process in both GAs. This selection method always produces the individual with the highest rating (utility). First, we compute the Utility Function (FU) for each chromosome (composition) using the utility function defined by the equation IV.3. Chromosomes are then classified according to their utilities. For the children's offspring, the two best parents are selected.

Crossover Operation:

Crossovers are a major player in generating new individuals. It is possible to exchange parts of both parents with some probability. We apply the gene segment crossover method. In the first GA, we randomly choose an intersection point (an integer between 1 and 9) in each service that makes up the two parents. Then services segments are exchanged two by two between the parents according to the crossing point. However, in the second GA, we will consider a service as a gene, and the crossing point will be chosen randomly between one and m (m is the number of services to be composed). Then, the two segments of both parents are swapped.

Mutation Operation:

Mutation is also a method that allows the creation of new individuals by altering one or more genes on a chromosome. In the first GA, mutations are applied to three randomly selected genes (three attributes) for each service in the theoretical composition. Since the goal is to maximize the utility function, we proceed as follows: If the attribute (gene) chosen for mutation is a positive attribute, replace its value with another value randomly generated between the current value and the maximum value for this attribute in the abstract service. On the other hand, if the selected attribute is negative, replace its value with a randomly generated value between the minimum and current value of this attribute in the abstract service.

In the second GA, each gene is a concrete service with its utility. To make the mutation, we also choose randomly a service and replace it with another among the concrete services belonging to the same abstract service.

To verify the feasibility and effectiveness of the proposed method in a large-scale environment, we perform five simulation experiments, as shown in the following subsections.

V.3.2 Exp 1: GNN-QSC approach behavior with respect to the QoS-levels number

In this experiment, the approach (GNN-QSC) will be evaluated according to the number of QoS-levels. For this, assuming that there are three abstract services in the composition plan ($m = 3$). Each abstract service contains thirty (n=30) concrete services. The execution time values for the algorithm are obtained from various values of M/ $M \in \{5, 10, 15, 20, 25, 30\}$. $M \ll n$ .Test results are shown in figure V.1. The vertical axis represents the execution time of the algorithm and the horizontal axis represents the M values.

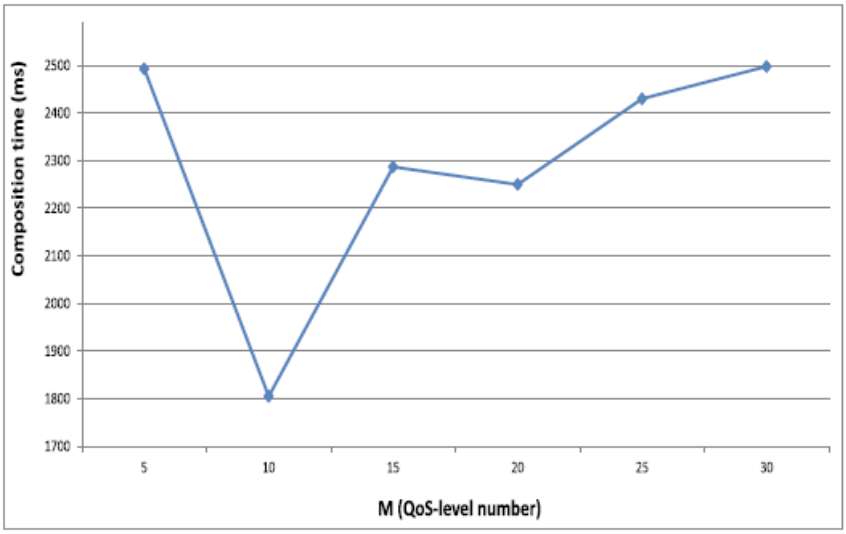

Figure V.1: GNN-QSC execution time with different QoS-levels.

Discussion:

As the figure V.1 shows, increasing the number of QoS levels M does not significantly increase (almost the same) the execution time of the service composition. Thus, the two large values of 2.493 and 2.498 seconds are obtained when M is equal to 5 and 30 respectively, and the smallest value of 1.805 seconds is obtained when $M = 10$, which is the best execution time. So set M to 10 in the next experiments.

V.3.3 Exp 2: GNN-QSC execution time evaluation

The purpose of this experiment is to examine the effect of the number of tasks to be composed on the performance of the composition process. To do this, set M to 10 (QoS levels), n to 40 (number of concrete services in each AS), and the number of abstract services (m) vary between 3 and 7. Compare the obtained results (running time) of our approach with those of the optimal compositions. The test results are shown in figure V.2.

Discussion:

As shown in the Figure V.2, the running time of GNN-QSC increases with the number of tasks number to be composed. Increases from 2.215 seconds to 8.325 seconds. This increase is very small, but acceptable compared to the optimal composition approach, which was exponential. Figure V.2 clearly shows the efficiency of GNN-QSC, significantly

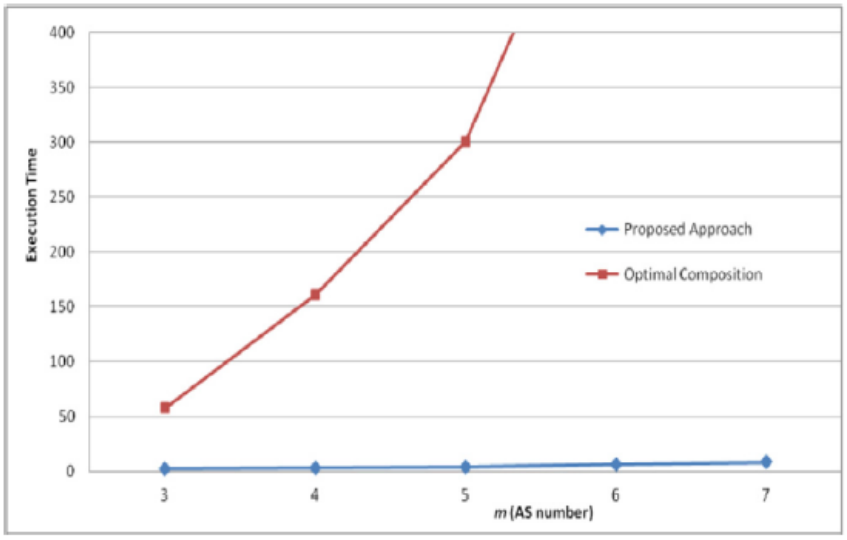

Figure V.2: GNN-QSC execution time with different number of AS.

reducing service composition time.

V.3.4 Exp 3: GNN-QSC execution time comparison with others approaches

In order to assess the performance and efficiency of GNN-QSC approach, simulation scenarios are carried out to compare it with five approaches. These latter are: the Two-Step QoS service Composition Algorithm (TS-QCA), approach using shuffled Frog Leaping Algorithm (SC-FLA); the Pareto-based partial Services selection Approach (PSA) [202], the Improved binary coded Genetic Algorithm (Improved GA) [195], and Multi-population Genetic Algorithm (MGA) [194]. The comparison indicator is the composition time, i.e. the necessary time to find the near-to-optimal composition. For this, we assume that there are three (03) abstract services in the compositional plane ($m = 3$), and $n \in \{20, 30, 40, 50, 60, 100, 200\}$. Then, we record the execution times average of fifty simulations for each approach. The results are shown in Figure V.3.

Discussion:

Figure V.3 shows that the composition time of both the GNN-QSC and TS-QCA algorithms does not increase significantly as the number of concrete services within each abstract service increases. In general, SC-FLA has stable composition time, in the large-scale environment. It has the lowest running time, followed by GNN-QSC. However, other approaches (PSA, MGA, and improved GA) significantly increase the composition time as the number of CSs increases. Overall, the GNN-QSC approach has the best composition time compared to the other approaches. This is due to the efficiency of the neural network model used as a method of weeding out unpromising services. This reduces the composition space and consequently the composition time.

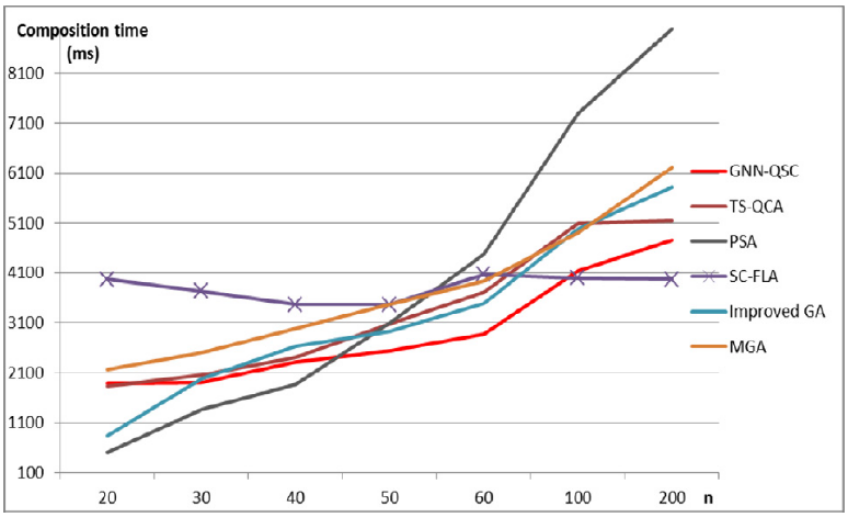

Figure V.3: Impact of the CS's number on the execution time.

V.3.5 Exp 4: GNN-QSC approach Hypervolume indicator evaluation

In this experiment, we compare the GNN-QSC approach with TS-QCA, SC-FLA, Improved GA, and MGA approaches. The comparative indicator is the Hypervolume (HV) indicator. The HV indicator is the most widely used indicator among several to compare the performance of multi-objective evolutionary algorithms [217, 218]. HV is a unary metric that computes the volume of the area enclosed by the solution set and a reference point, with higher values indicating better results. The author of [219] claims that HV covers accuracy, diversity and cardinality as HV is the only unary metric with this feature. Figure V.4 shows the comparison result.

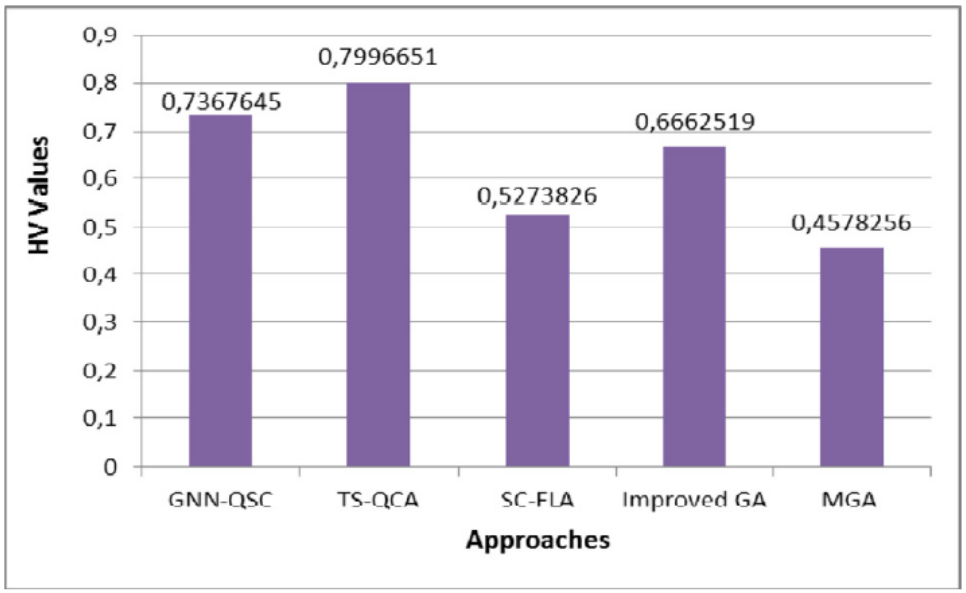

Figure V.4: Hypervolume indicator comparison.

Discussion:

From the figure V.4, we can see that the HV values of GNN-QSC and TS-QCA are almost the same, and TS-QCA is slightly better. This means that the solution sets for GNN-QSC and TS-QCA are nearly identical. However, the HV values confirm that GNN-QSC has a much better performance solution set compared to SC-FLA, Improved GA, and MGA approaches.

V.3.6 Exp 5: GNN-QSC approach optimality evaluation

In this final experiment, we compare the optimality of the composition of GNN-QSC, TS-QCA, SC-FLA, Improved GA, MGA and PSA algorithms. The optimality of a composition is the ratio of the value of the utility function for a composition that is close to optimal to its optimal value. Suppose the compositional plane has three abstract services ($m = 3$), $n \in \{20, 30, 40, 50, 60, 100, 200\}$. We then compute the average optimality of about fifty simulations for each approach. The result is shown in figure V.5.

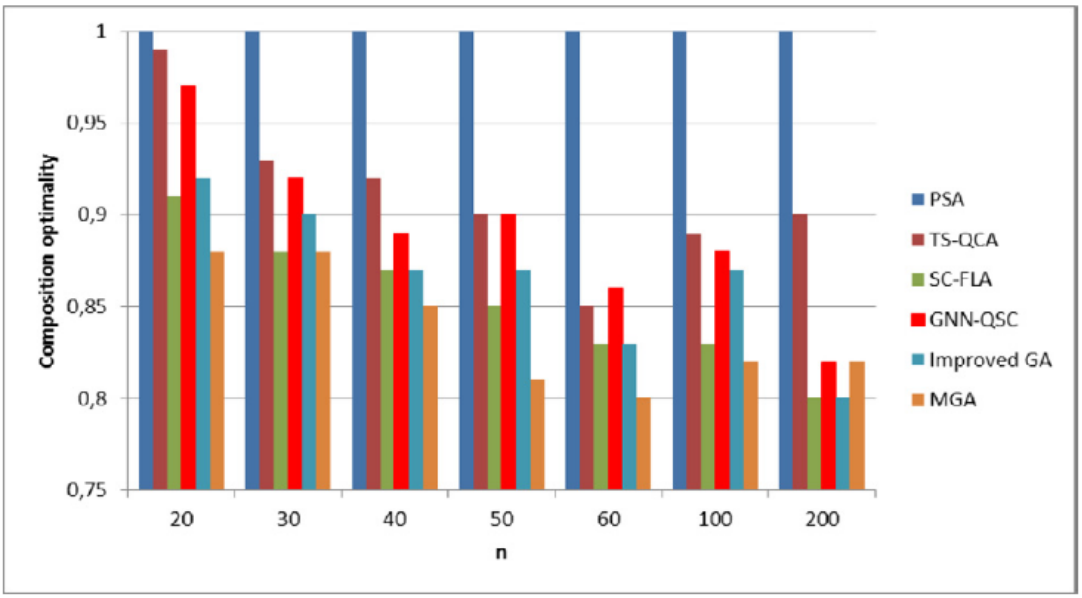

Figure V.5: Optimality against the concrete services number.

Discussion:

As can be seen in the figure V.5, the GNN-QSC algorithm outperforms the SC-FLA, Improved GA and MGA algorithms in terms of composition optimality. This is justified by using neural model. This allows us to focus only on services that make sense from a QoS perspective. However, the optimality of TS-QCA is slightly better than that of GNN-QSC. The optimality of the PSA algorithm is ideal because it uses the concept of Pareto dominance to find the optimal composition. However, GNN-QSC is a metaheuristic method based on genetic algorithms that finds near-optimal compositions in a very short time.

V.4 Conclusion

During this chapter, we presented an illustrative scenario showing in a clear way the sequence of steps of the approach proposed for the dynamic composition of IoT services under global QoS constraint. Then, through a set of experiments, an evaluation of the performance of said approach was made. The results show that the proposed approach considerably reduces the composition time, this is mainly due to the use of neural networks as a clustering technique, which will considerably reduce the composition space. The GNN-QSC approach reduces composition time without too much degrade the composition optimality, which is the strength of this approach compared to other approaches.

Conclusion and Perspectives

Synthesis

The rapid development of distributed information systems and the spread of Internet access have led to new paradigm development of interaction between applications. Thus, the component-based paradigms evolved into the Service-Oriented Architecture (SOA), which revolves around the concept of web service. The Internet of Things (IoT), being a new and growing technology allowing the interconnection and interaction between all types of objects (objects, animals and people, etc.) through the existing Internet infrastructure. By adopting the Service-Oriented Architecture in IoT, the functionalities of connected objects can be encapsulated as web services through the web, which we call IoT services. The important feature of IoT or web services is their property of composability which allows to create more complex services by combining more basic services taking into account user needs. The services composition involving the ability to select, coordinate, interact, and interoperate existing services is a complex task. This complexity is heightened when it comes to dynamically integrating services on demand, and automatically composing them to meet requirements that are not met by existing ones. During this thesis, we have focused our attention on the QoS-aware IoT services composition problem. Indeed, we believe that an approach for IoT services composition must offer the potential to realize flexible and adaptable applications, by selecting and combining the IoT services appropriately based on the user's request and preferences in terms of QoS. It is in this context that our thesis work takes place where an automatic composition approach for IoT services based on Artificial Intelligence (AI) techniques, mainly Neural Networks and the Genetic Algorithm, is proposed. The proposed approach considers the overall QoS optimization of the composite service while respecting the user preferences.

The first part of this thesis is reserved for state-of-the-art works, in which we presented the main standards related to embedded systems as a basic technology for the advent of the Internet of Things. Afterwards, a presentation of the concepts related to the Internet of Things is made. After the specification of the QoS-aware IoT services composition problem, we have developed a detailed state of the art flying over the different approaches that allow its resolution, namely: (1) bio-inspired meta-heuristic approaches, mainly based on the Genetic Algorithm and Particle Swarm Optimization; (2) approaches based on clustering techniques to reduce the composition space, mainly based on the K-means

technique; and (3) approaches focus rather on the energy aspect of the composition. The study and the analysis of the state of the art works made it possible on the one hand, to identify the advantages and the limits of each approach, and on the other hand, to better position our approach to see how it would be possible to improve performance criteria.

The second part is dedicated to the contributions where we described our proposal. Our approach models the "QoS-aware Automatic Service Composition" problem, which is NP-hard problem, as a process of predicting the category of atomic services constituting the composition. After a QoS intervals breakdown into QoS-levels, an Ideal Theoretical Composition (ITC) is sought on the basis of an initial population made up of theoretical services generated in the decomposition phase. The theoretical services constituting the ITC will be evaluated as well as the concrete services using a neural network in order to group them according to their QoS parameters. Based on this evaluation, we can predict the category of concrete services to be contracted for each abstract service, thus eliminating the other categories. The thing that will reduce the composition space and therefore the composition time. At this stage, the genetic algorithm is used to find the concrete composition that is almost optimal respecting the user's QoS preferences and guaranteeing a better global QoS. All the experiments carried out show the effectiveness of the proposed approach in terms of execution time, the Hypervolume indicator and the composition optimality.

Perspectives

Furthermore, we believe that our work opens up other avenues of research. Several extensions can be proposed. These perspectives respond to a main objective which is to improve the composition of semantic Web of Things services. The perspectives we present represent improvements of the proposed IoT service composition approach.

- *Functional properties:* The consideration of non-functional properties that we have adopted is a simple representation that covers a wide range of non-functional properties that can be expressed as simple constraints. We plan to combine functional properties with non-functional properties in the IoT services composition.
- *Use of other composition structures:* we also plan to extend the performance evaluation of the proposed service composition approach, in particular by considering other more complex composition structures (conditional, loops)
- *Context awareness:* we also intend to enrich IoT service descriptions with contextual information that is crucial in service-oriented systems. Indeed, contextual information (such as, for example, the user's center of attention, his location, his orientation, the date and time when he evolves, etc.) play an important role in the personalization of the services composition to provide a satisfactory result to the user.
- Finally, we intend to test our composition approach on more complex scenarios and with concrete IoT services and real users.

Bibliography

[1] Mark Weiser. The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review*, 3(3):3–11, 1999.

[2] Sylvain Cherrier and Rami Langar. Services organisation in iot: mixing orchestration and choreography. In *2018 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–4. IEEE, 2018.

[3] Rabah Boucetti, Sofiane Mounine Hemam, and Ouassila Hioual. Multicriteria approach for discovering iot services. In *International Conference on Computing Systems and Applications*, pages 208–219. Springer, 2020.

[4] Nebil Ben Mabrouk, Nikolaos Georgantas, and Valerie Issarny. Set-based bi-level optimisation for qos-aware service composition in ubiquitous environments. In *2015 IEEE International Conference on Web Services*, pages 25–32. IEEE, 2015.

[5] Seyedsalar Sefati and Nima Jafari Navimipour. A qos-aware service composition mechanism in the internet of things using a hidden-markov-model-based optimization algorithm. *IEEE Internet of Things Journal*, 8(20):15620–15627, 2021.

[6] Qiping She, Xiaochao Wei, Guihua Nie, and Donglin Chen. Qos-aware cloud service composition: A systematic mapping study from the perspective of computational intelligence. *Expert Systems with Applications*, 138:112804, 2019.

[7] Arun Kumar Sangaiah, Gui-Bin Bian, Seyed Mostafa Bozorgi, Mohsen Yaghoubi Suraki, Ali Asghar Rahmani Hosseinabadi, and Morteza Babazadeh Shareh. A novel quality-of-service-aware web services composition using biogeography-based optimization algorithm. *Soft Computing*, 24(11):8125–8137, 2020.

[8] Mohamed Essaid Khanouche, Ferhat Attal, Yacine Amirat, Abdelghani Chibani, and Moussa Kerkar. Clustering-based and qos-aware services composition algorithm for ambient intelligence. *Information Sciences*, 482:419–439, 2019.

[9] Hadi Naghavipour, Tey Kok Soon, Mohd Yamani Idna Idris, Morteza Namvar, Rosli Bin Salleh, and Abdullah Gani. Hybrid metaheuristics for qos-aware service composition: A systematic mapping study. *IEEE Access*, 2021.

[10] Amal Kouicem, Mohamed Essaid Khanouche, and Abdelkamel Tari. Novel bat algorithm for qos-aware services composition in large scale internet of things. *Cluster Computing*, pages 1–15, 2022.

[11] Yuhong Yan, Min Chen, and Yubin Yang. Anytime qos optimization over the plangraph for web service composition. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 1968–1975, 2012.

[12] Parvaneh Asghari, Amir Masoud Rahmani, and Hamid Haj Seyyed Javadi. Service composition approaches in iot: A systematic review. *Journal of Network and Computer Applications*, 120:61–77, 2018.

[13] Peter Marwedel. *Embedded system design: embedded systems foundations of cyber-physical systems, and the internet of things*. Springer Nature, 2021.

[14] Francine Krief. Les systèmes embarqués communicants: Mobilité, sécurité, autonomie, 2008.

[15] Peter Marwedel. *Embedded System Design*. Kluwer Academic Publishers, 2003.

[16] Michael Barr. Embedded systems glossary. *Neutrino Technical Library*, 2007.

[17] V Ramya and B Palaniappan. Embedded system for hazardous gas detection and alerting. *International Journal of Distributed and Parallel Systems (IJDPS)*, 3(3):287–300, 2012.

[18] PM Sagar and Vivek Agarwal. Embedded operating systems for real-time applications. In *M. Tech. credit seminar report, Electronic Systems Group, EE Dept, IIT Bombay, Submitted in November*, 2002.

[19] Mahmood Ashraf and Masitah Ghazali. Analysis of physicality aspects in physical user interfaces of embedded systems. *Jurnal Teknologi*, 61(1), 2013.

[20] Brock J LaMeres. Introduction to embedded systems. In *Embedded Systems Design using the MSP430FR2355 LaunchPad™*, pages 1–6. Springer, 2020.

[21] Chafik ARAR. *Redondance logicielle pour la tolérance aux fautes des communications*. PhD thesis, Université de Batna 2, 2016.

[22] Ali Abbasi, Jos Wetzels, Thorsten Holz, and Sandro Etalle. Challenges in designing exploit mitigations for deeply embedded systems. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 31–46. IEEE, 2019.

[23] B MADHU SUDHAN REDDY, B MANASA, and B MAHENDAR REDDY. Wireless power transmission for embedded systems. *International Journal of Scientific Engineering and Technology Research*, 2013.

[24] Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic timing analysis techniques for real-time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–60, 2019.

[25] Pascal Chevochot and Isabelle Puaut. An approach for fault-tolerance in hard real-time distributed systems. In *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, pages 292–293. IEEE, 1999.

[26] Abdelaziz Kara. *Le développement d'un système embarqué implémentant le Peer-to-Peer pour la TVoIP*. PhD thesis, Université FERHAT ABBAS SETIF, 2018.

[27] Raj Kamal. *Embedded systems: architecture, programming and design*. Tata McGraw-Hill Education, 2011.

[28] Steve Heath. *Embedded systems design*. Elsevier, 2002.

[29] Rob Toulson and Tim Wilmshurst. *Fast and effective embedded systems design: applying the ARM mbed*. Elsevier, 2012.

[30] Toshiki Kanamoto, Masami Fukushima, Koichi Kitagishi, Seijin Nakayama, Hideki Ishihara, Koki Kasai, Atsushi Kurokawa, and Masashi Imai. A single-stage risc-v processor to mitigate the von neumann bottleneck. In *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1085–1088. IEEE, 2019.

[31] Carsten Heinz, Yannick Lavan, Jaco Hofmann, and Andreas Koch. A catalog and in-hardware evaluation of open-source drop-in compatible risc-v softcore processors. In *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pages 1–8. IEEE, 2019.

[32] Ram Kumar, Akhilesh Singhania, Andrew Castner, Eddie Kohler, and Mani Srivastava. A system for coarse grained memory protection in tiny embedded processors. In *2007 44th ACM/IEEE Design Automation Conference*, pages 218–223. IEEE, 2007.

[33] Aastik Upadhyay and Abhimanyu S Dhapola. Embedded systems and its application in medical field. *Emerging Trends in Computer Science and Information*, 3, 2015.

[34] Andrew Tanenbaum. *Modern operating systems*. Pearson Education, Inc.,, 2009.

[35] S Dharshana, Vidya Krishnan, and VC Divya. Therapeutic applications of ultrasound in dentistry. *Journal of Pharmaceutical Sciences and Research*, 12(11):1394–1399, 2020.

[36] W Wong-Ng. Ceramic sensors. *International Tables for Crystallography*, 2019.

[37] Xing Liu and Orlando Baiocchi. A comparison of the definitions for smart sensors, smart objects and things in iot. In *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 1–4. IEEE, 2016.

[38] Ammar Rayes and Samer Salam. *Internet of things from hype to reality*. Springer, 2019.

[39] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, Joseph Walsh, et al. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6):2140, 2021.

[40] IN MODERN. New trends in smart sensors for industrial applications—part i. *IEEE Transactions on Industrial Electronics*, 64(9):7281, 2017.

[41] Martin Pech, Jaroslav Vrchota, and Jiří Bednář. Predictive maintenance and intelligent sensors in smart factory. *Sensors*, 21(4):1470, 2021.

[42] Raluca Maria Aileni. Cloud hybrid service for monitoring building energy efficiency obtained by using insulation structures. *on Virtual Learning*, page 342, 2015.

[43] Burcu Arman Kuzubasoglu and Senem Kursun Bahadir. Flexible temperature sensors: A review. *Sensors and Actuators A: Physical*, 315:112282, 2020.

[44] Fenlan Xu, Xiuyan Li, Yue Shi, Luhai Li, Wei Wang, Liang He, and Ruping Liu. Recent developments for flexible pressure sensors: A review. *Micromachines*, 9(11):580, 2018.

[45] Gonzalo Farias, Ernesto Fabregas, Emmanuel Peralta, Héctor Vargas, Gabriel Hermosilla, Gonzalo Garcia, and Sebastián Dormido. A neural network approach for building an obstacle detection model by fusion of proximity sensors data. *Sensors*, 18(3):683, 2018.

[46] B Ragavi, L Pavithra, P Sandhiyadevi, GK Mohanapriya, and S Harikirubha. Smart agriculture with ai sensor by using agrobot. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1–4. IEEE, 2020.

[47] AG Correa-Mena, LA González, LJ Quintero-Rodríguez, and IE Zaldivar-Huerta. Review on integrated optical sensors and its applications. In *2017 IEEE Mexican Humanitarian Technology Conference (MHTC)*, pages 170–173. IEEE, 2017.

[48] Jun Ohta. *Smart CMOS image sensors and applications*. CRC press, 2017.

[49] Carlo Corsi. History highlights and future trends of infrared sensors. *Journal of modern optics*, 57(18):1663–1686, 2010.

[50] Hamid Farahani, Rahman Wagiran, and Mohd Nizar Hamidon. Humidity sensors principle, mechanism, and fabrication technologies: a comprehensive review. *Sensors*, 14(5):7881–7939, 2014.

[51] Judicaël Picaut, Arnaud Can, Nicolas Fortin, Jeremy Ardouin, and Mathieu Lagrange. Low-cost sensors for urban noise monitoring networks—a literature review. *Sensors*, 20(8):2256, 2020.

[52] Hartmut Janocha. *Actuators*. Springer, 2004.

[53] Ahmad H AbdulKarim, Mohsen A Tawfik, Ahmed F Hasan, and Ahmed TY El-Awady. Review of improving energy efficiency technologies. *Journal of Environmental Science. Ain Shams*, 50(8):239–286, 2021.

[54] Bernard De Fornel and Jean-Paul Louis. *Electrical Actuators: Applications and Performance*. John Wiley & Sons, 2013.

[55] Michaël De Volder and Dominiek Reynaerts. Pneumatic and hydraulic microactuators: a review. *Journal of Micromechanics and microengineering*, 20(4):043001, 2010.

[56] Hazem I Ali, SBBM Noor, SM Bashi, and MH Marhaban. A review of pneumatic actuators (modeling and control). *Australian Journal of Basic and Applied Sciences*, 3(2):440–454, 2009.

[57] Massimo Pasquale. Mechanical sensors and actuators. *Sensors and Actuators A: Physical*, 106(1-3):142–148, 2003.

[58] Bill Glover and Himanshu Bhatt. *RFID essentials*. " O'Reilly Media, Inc.", 2006.

[59] Xiaolin Jia, Quanyuan Feng, Taihua Fan, and Quanshui Lei. Rfid technology and its applications in internet of things (iot). In *2012 2nd international conference on consumer electronics, communications and networks (CECNet)*, pages 1282–1285. IEEE, 2012.

[60] P Kumar, HW Reinitz, J Simunovic, KP Sandeep, and PD Franzon. Overview of rfid technology and its applications in the food industry. *Journal of Food Science*, 74(8):R101–R106, 2009.

[61] Badri Nath, Franklin Reynolds, and Roy Want. Rfid technology and applications. *IEEE Pervasive computing*, 5(1):22–24, 2006.

[62] Arun N Nambiar. Rfid technology: A review of its applications. In *Proceedings of the world congress on engineering and computer science*, volume 2, pages 20–22. Citeseer, 2009.

[63] Imad Saleh. Internet des objets (ido): Concepts, enjeux, défis et perspectives. *Revue Internet des objets*, 2(10.21494), 2018.

[64] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[65] Aarti Rao Jaladi, Karishma Khithani, Pankaja Pawar, Kiran Malvi, and Gauri Sahoo. Environmental monitoring using wireless sensor networks (wsn) based on iot. *Int. Res. J. Eng. Technol*, 4(1):1371–1378, 2017.

[66] Hossein Shahinzadeh, Jalal Moradi, Gevork B Gharehpetian, Hamed Nafisi, and Mehrdad Abedi. Iot architecture for smart grids. In *2019 International Conference on Protection and Automation of Power System (IPAPS)*, pages 22–30. IEEE, 2019.

[67] Matteo Petracca, Stefano Bocchino, Andrea Azzarà, Riccardo Pelliccia, Marco Ghibaudi, and Paolo Pagano. Wsn and rfid integration in the iot scenario: an advanced safety system for industrial plants. *Journal of communications software and systems*, 9(1):104–113, 2013.

[68] Harika Devi Kotha and V Mnssvkr Gupta. Iot application: a survey. *Int. J. Eng. Technol*, 7(2.7):891–896, 2018.

[69] Roberto Minerva, Abyi Biru, and Domenico Rotondi. Towards a definition of the internet of things (iot). *IEEE Internet Initiative*, 1(1):1–86, 2015.

[70] Hakima Chaouchi. *The internet of things: Connecting objects to the web*. John Wiley & Sons, 2013.

[71] Dan Wang, Dong Chen, Bin Song, Nadra Guizani, Xiaoyan Yu, and Xiaojiang Du. From iot to 5g i-iot: The next generation iot-based intelligent algorithms and 5g technologies. *IEEE Communications Magazine*, 56(10):114–120, 2018.

[72] D. Evans. *L'Internet des objets: Comment l'évolution actuelle d'Internet transforme-t-elle le monde ?* Livre blanc. Cisco Internet Busi ness Solutions Group (IBSG), 2011.

[73] Mustafa Kocakulak and Ismail Butun. An overview of wireless sensor networks towards internet of things. In *2017 IEEE 7th annual computing and communication workshop and conference (CCWC)*, pages 1–6. Ieee, 2017.

[74] Mudassar Ahmad, Atiab Ishtiaq, Muhammad Asif Habib, and Syed Hassan Ahmed. A review of internet of things (iot) connectivity techniques. *Recent trends and advances in wireless and IoT-enabled networks*, pages 25–36, 2019.

[75] Tobias Pfandzelter and David Bermbach. Iot data processing in the fog: Functions, streams, or batch processing? In *2019 IEEE International conference on fog computing (ICFC)*, pages 201–206. IEEE, 2019.

[76] SP Sasirekha, A Priya, T Anita, and P Sherubha. Data processing and management in iot and wireless sensor network. In *Journal of Physics: Conference Series*, volume 1712, page 012002. IOP Publishing, 2020.

[77] Toby Jia-Jun Li, Yuanchun Li, Fanglin Chen, and Brad A Myers. Programming iot devices by demonstration using mobile apps. In *International Symposium on End User Development*, pages 3–17. Springer, 2017.

[78] Ilan Kirsh and Heinrich Ruser. Phone-pointing remote app: Using smartphones as pointers in gesture-based iot remote controls. In *International Conference on Human-Computer Interaction*, pages 14–21. Springer, 2021.

[79] Mohammad Ali Jabraeil Jamali, Bahareh Bahrami, Arash Heidari, Parisa Allahverdizadeh, and Farhad Norouzi. Iot architecture. *Towards the Internet of Things*, pages 9–31, 2020.

[80] NM Masoodhu Banu and C Sujatha. Iot architecture a comparative study. *Int J Pur Appl Math*, 117(8):45–49, 2017.

[81] Sarah A Al-Qaseemi, Hajer A Almulhim, Maria F Almulhim, and Saqib Rasool Chaudhry. Iot architecture challenges and issues: Lack of standardization. In *2016 Future technologies conference (FTC)*, pages 731–738. IEEE, 2016.

[82] Anjum Khairi, M Farooq, M Waseem, and S Mazhar. A critical analysis on the security concerns of internet of things (iot). *Perception*, 111, 2015.

[83] Amit Kumar Sikder, Giuseppe Petracca, Hidayet Aksu, Trent Jaeger, and A Selcuk Uluagac. A survey on sensor-based threats to internet-of-things (iot) devices and applications. *arXiv preprint arXiv:1802.02041*, 2018.

[84] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Future internet: the internet of things architecture, possible applications and key challenges. In *2012 10th international conference on frontiers of information technology*, pages 257–260. IEEE, 2012.

[85] Keyur K Patel, Sunil M Patel, et al. Internet of things-iot: definition, characteristics, architecture, enabling technologies, application & future challenges. *International journal of engineering science and computing*, 6(5), 2016.

[86] Ovidiu Vermesan, Peter Friess, et al. *Internet of things-from research and innovation to market deployment*, volume 29. River publishers Aalborg, 2014.

[87] Emmanuel Amiot. Internet des objets: les business models remis en cause. Technical report, Technical report, Oliver Wyman, 2015.

[88] Radouan Ait Radouan Ait Mouha et al. Internet of things (iot). *Journal of Data Analysis and Information Processing*, 9(02):77, 2021.

[89] Piers Barrios, Christophe Danjou, and Benoit Eynard. Literature review and methodological framework for integration of iot and plm in manufacturing industry. *Computers in Industry*, 140:103688, 2022.

[90] Tahera Kalsoom, Shehzad Ahmed, Piyya Muhammad Rafi-ul Shan, Muhammad Azmat, Pervaiz Akhtar, Zeeshan Pervez, Muhammad Ali Imran, and Masood Ur-Rehman. Impact of iot on manufacturing industry 4.0: A new triangular systematic review. *Sustainability*, 13(22):12506, 2021.

[91] S Muthuramalingam, A Bharathi, N Gayathri, R Sathiyaraj, B Balamurugan, et al. Iot based intelligent transportation system (iot-its) for global perspective: A case study. In *Internet of Things and Big Data Analytics for Smart Generation*, pages 279–300. Springer, 2019.

[92] H Varun Chand and J Karthikeyan. Survey on the role of iot in intelligent transportation system. *Indonesian Journal of Electrical Engineering and Computer Science*, 11(3):936–941, 2018.

[93] Yanxing Song, F Richard Yu, Li Zhou, Xi Yang, and Zefang He. Applications of the internet of things (iot) in smart logistics: a comprehensive survey. *IEEE Internet of Things Journal*, 8(6):4250–4274, 2020.

[94] A Martín-Garín, JA Millán-García, A Baïri, M Gabilondo, and A Rodríguez. Iot and cloud computing for building energy efficiency. In *Start-Up Creation*, pages 235–265. Elsevier, 2020.

[95] Wenbo Wang, Haidong Yang, Yingfeng Zhang, and Jianxue Xu. Iot-enabled real-time energy efficiency optimisation method for energy-intensive manufacturing enterprises. *International Journal of Computer Integrated Manufacturing*, 31(4-5):362–379, 2018.

[96] Eljona Zanaj, Giuseppe Caso, Luca De Nardis, Alireza Mohammadpour, Özgü Alay, and Maria-Gabriella Di Benedetto. Energy efficiency in short and wide-area iot technologies—a survey. *Technologies*, 9(1):22, 2021.

[97] Chrysi K Metallidou, Kostas E Psannis, and Eugenia Alexandropoulou Egyptiadou. Energy efficiency in smart buildings: Iot approaches. *IEEE Access*, 8:63679–63699, 2020.

[98] Md Shakawat Hossain, Nur Mohammad Ali Chisty, Donyea Lamont Hargrove, and Ruhul Amin. Role of internet of things (iot) in retail business and enabling smart retailing experiences. *Asian Business Review*, 11(2):75–80, 2021.

[99] Neha Mishra and Anup Kumar Keshri. Smart racking and retailing using iot. In *Proceedings of the Fourth International Conference on Microelectronics, Computing and Communication Systems*, pages 645–653. Springer, 2021.

[100] Felipe Caro and Ramin Sadr. The internet of things (iot) in retail: Bridging supply and demand. *Business Horizons*, 62(1):47–54, 2019.

[101] Yong Kyu Lee. Review of the role of the internet of things (iot) on the consumer market: Focusing on smart tourism, healthcare, and retailing. *Examining the Socio-Technical Impact of Smart Cities*, pages 180–198, 2021.

[102] Rondik J Hassan, SR Zeebaree, Siddeeq Y Ameen, Shakir Fattah Kak, MA Sadeeq, Zainab Salih Ageed, AZ Adel, and Azar Abid Salih. State of art survey for iot effects on smart city technology: challenges, opportunities, and solutions. *Asian Journal of Research in Computer Science*, 22:32–48, 2021.

[103] Zhao Tong, Feng Ye, Ming Yan, Hong Liu, and Sunitha Basodi. A survey on algorithms for intelligent computing and smart city applications. *Big Data Mining and Analytics*, 4(3):155–172, 2021.

[104] Nabaa Ali Jasim, Haider TH, and Salim AL Rikabi. Design and implementation of smart city applications based on the internet of things. *International Journal of Interactive Mobile Technologies*, 15(13), 2021.

[105] Mohamed Yousif, Chaminda Hewage, and Liqaa Nawaf. Iot technologies during and beyond covid-19: a comprehensive review. *Future Internet*, 13(5):105, 2021.

[106] Mohd Javaid and Ibrahim Haleem Khan. Internet of things (iot) enabled healthcare helps to take the challenges of covid-19 pandemic. *Journal of Oral Biology and Craniofacial Research*, 11(2):209–214, 2021.

[107] Priyanka Dwivedi and Monoj Kumar Singha. Iot based wearable healthcare system: Post covid-19. In *The Impact of the COVID-19 Pandemic on Green Societies*, pages 305–321. Springer, 2021.

[108] Pushan Kumar Dutta and Susanta Mitra. Application of agricultural drones and iot to understand food supply chain during post covid-19. *Agricultural Informatics: Automation Using the IoT and Machine Learning*, pages 67–87, 2021.

[109] Tharaka de Vass, Himanshu Shee, and Shah Miah. Iot in supply chain management: opportunities and challenges for businesses in early industry 4.0 context. *Operations and Supply Chain Management: An International Journal*, 14(2):148–161, 2021.

[110] Weng Chun Tan and Manjit Singh Sidhu. Review of rfid and iot integration in supply chain management. *Operations Research Perspectives*, page 100229, 2022.

[111] Veerachamy Ramachandran, Ramar Ramalakshmi, Balasubramanian Prabhu Kavin, Irshad Hussain, Abdulrazak H Almaliki, Abdulrhman A Almaliki, Ashraf Y Elnaggar, and Enas E Hussein. Exploiting iot and its enabled technologies for irrigation needs in agriculture. *Water*, 14(5):719, 2022.

[112] Bam Bahadur Sinha and R Dhanalakshmi. Recent advancements and challenges of internet of things in smart agriculture: A survey. *Future Generation Computer Systems*, 126:169–184, 2022.

[113] Wahiba Yaïci, Karthik Krishnamurthy, Evgueniy Entchev, and Michela Longo. Recent advances in internet of things (iot) infrastructures for building energy systems: A review. *Sensors*, 21(6):2152, 2021.

[114] Wonyoung Choi, Jisu Kim, SangEun Lee, and Eunil Park. Smart home and internet of things: A bibliometric study. *Journal of Cleaner Production*, 301:126908, 2021.

[115] Cristina Stolojescu-Crisan, Calin Crisan, and Bogdan-Petru Butunoi. An iot-based smart home automation system. *Sensors*, 21(11):3784, 2021.

[116] Maria Stoyanova, Yannis Nikoloudakis, Spyridon Panagiotakis, Evangelos Pallis, and Evangelos K Markakis. A survey on the internet of things (iot) forensics: challenges, approaches, and open issues. *IEEE Communications Surveys & Tutorials*, 22(2):1191–1221, 2020.

[117] Hamed HaddadPajouh, Ali Dehghantanha, Reza M Parizi, Mohammed Aledhari, and Hadis Karimipour. A survey on internet of things security: Requirements, challenges, and solutions. *Internet of Things*, 14:100129, 2021.

[118] Anh Nguyen Duc, Ronald Jabangwe, Pangkaj Paul, and Pekka Abrahamsson. Security challenges in iot development: a software engineering perspective. In *Proceedings of the XP2017 scientific workshops*, pages 1–5, 2017.

[119] Chintan Patel and Nishant Doshi. Security challenges in iot cyber world. In *Security in Smart Cities: Models, Applications, and Challenges*, pages 171–191. Springer, 2019.

[120] Teemu Savolainen, Jonne Soininen, and Bilhanan Silverajan. Ipv6 addressing strategies for iot. *IEEE Sensors Journal*, 13(10):3511–3519, 2013.

[121] CC Sobin. A survey on architecture, protocols and challenges in iot. *Wireless Personal Communications*, 112(3):1383–1429, 2020.

[122] Kyriakos Georgiou, Samuel Xavier-de Souza, and Kerstin Eder. The iot energy challenge: A software perspective. *IEEE Embedded Systems Letters*, 10(3):53–56, 2017.

[123] Mobark Q Aldossari and Anna Sidorova. Consumer acceptance of internet of things (iot): Smart home context. *Journal of Computer Information Systems*, 60(6):507–517, 2020.

[124] Michael Onuoha Thomas, Beverly Amunga Onyimbo, and Rajasvaran Logeswaran. Usability evaluation criteria for internet of things. *Int J Inf Technol Comput Sci*, 8:10–18, 2016.

[125] Rossana MC Andrade, Belmondo R Aragão Junior, Pedro Almir M Oliveira, Marcio EF Maia, Windson Viana, and Tales P Nogueira. Multifaceted infrastructure for self-adaptive iot systems. *Information and Software Technology*, 132:106505, 2021.

[126] I Supriana, K Surendro, H Mubarok, et al. Self-adaptation modeling for service evolution on the internet of things (iot). In *IOP Conference Series: Materials Science and Engineering*, volume 550, page 012026. IOP Publishing, 2019.

[127] Mohammad Asad Abbasi, Zulfiqar A Memon, Nouman M Durrani, Waleej Haider, Kashif Laeeq, and Ghulam Ali Mallah. A multi-layer trust-based middleware framework for handling interoperability issues in heterogeneous iots. *Cluster Computing*, 24(3):2133–2160, 2021.

[128] Andrea Cimmino, María Poveda-Villalón, and Raúl García-Castro. ewot: A semantic interoperability approach for heterogeneous iot ecosystems based on the web of things. *Sensors*, 20(3):822, 2020.

[129] Tara Salman and Raj Jain. A survey of protocols and standards for internet of things. *arXiv preprint arXiv:1903.11549*, 2019.

[130] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)*, 51(6):1–29, 2019.

[131] Miljan Sikimić, Momčilo Amović, Vladimir Vujović, Bojan Suknović, and Dragan Manjak. An overview of wireless technologies for iot network. In *2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–6. IEEE, 2020.

[132] Naresh Kumar Gupta. *Inside Bluetooth low energy*. Artech House, 2016.

[133] Jonathan Fürst, Kaifei Chen, Hyung-Sin Kim, and Philippe Bonnet. Evaluating bluetooth low energy for iot. In *2018 IEEE Workshop on Benchmarking Cyber-Physical Networks and Systems (CPSBench)*, pages 1–6. IEEE, 2018.

[134] B Swathi and Yerrolla Chanti. Review on simplifying iot the usage of near field communication (nfc) in digital gadget. *journal of mechanics of continua and mathematical sciences (jmcms)*, 15:464–474, 2020.

[135] Hossein Pirayesh, Pedram Kheirkhah Sangdeh, and Huacheng Zeng. Coexistence of wi-fi and iot communications in wlans. *IEEE Internet of Things Journal*, 7(8):7495–7505, 2020.

[136] PA Savale. A comparative study of 1g to 5g generations in the wireless mobile technology: A review. *Advances in Computer Science and Information Technology*, page 7, 2020.

[137] Oratile Khutsoane, Bassey Isong, and Adnan M Abu-Mahfouz. Iot devices and applications based on lora/lorawan. In *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*, pages 6107–6112. IEEE, 2017.

[138] Matthieu Kanj, Vincent Savaux, and Mathieu Le Guen. A tutorial on nb-iot physical layer design. *IEEE Communications Surveys & Tutorials*, 22(4):2408–2446, 2020.

[139] Ayanle I Ali, Sibel Zorlu Partal, Salih Kepke, and Hakan P Partal. Zigbee and lora based wireless sensors for smart environment and iot applications. In *2019 1st Global Power, Energy and Communication Conference (GPECOM)*, pages 19–23. IEEE, 2019.

[140] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. Overview of cellular lpwan technologies for iot deployment: Sigfox, lorawan, and nb-iot. In *2018 ieee international conference on pervasive computing and communications workshops (percom workshops)*, pages 197–202. IEEE, 2018.

[141] Zhi Yong Luo, Qing Wang, and Ting Cai. Sensor node identifier resolution of ip-based and non-ip-based wsn. In *Applied Mechanics and Materials*, volume 511, pages 154–157. Trans Tech Publ, 2014.

[142] Zach Shelby and Carsten Bormann. *6LoWPAN: The wireless embedded Internet*. John Wiley & Sons, 2011.

[143] Rolando Herrero. *Fundamentals of IoT Communication Technologies*. Springer, 2022.

[144] Mads Lauridsen, Lucas Chavarria Gimenez, Ignacio Rodriguez, Troels B Sorensen, and Preben Mogensen. From lte to 5g for connected mobility. *IEEE Communications Magazine*, 55(3):156–162, 2017.

[145] Nelson C Almeida, Rodrigo P Rolle, Eduardo P Godoy, Paolo Ferrari, and Emiliano Sisinni. Proposal of a hybrid lora mesh/lorawan network. In *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, pages 702–707. Ieee, 2020.

[146] Alexandru Lavric, Adrian I Petrariu, and Valentin Popa. Sigfox communication protocol: The new era of iot? In *2019 International Conference on Sensing and Instrumentation in IoT Era (ISSI)*, pages 1–4. IEEE, 2019.

[147] Abdullah Ahmed Omar Bahashwan and Selvakumar Manickam. A brief review of messaging protocol standards for internet of things (iot). *Journal of Cyber Security and Mobility*, pages 1–14, 2019.

[148] Pavel Masek, Jiri Hosek, Krystof Zeman, Martin Stusek, Dominik Kovac, Petr Cika, Jan Masek, Sergey Andreev, and Franz Kröpfl. Implementation of true iot vision: survey on enabling protocols and hands-on experience. *International Journal of Distributed Sensor Networks*, 12(4):8160282, 2016.

[149] Yuya Sasaki and Tetsuya Yokotani. Performance evaluation of mqtt as a communication protocol for iot and prototyping. *Advances in Technology Innovation*, 4(1):21, 2019.

[150] Biswajeeban Mishra and Attila Kertesz. The use of mqtt in m2m and iot systems: A survey. *IEEE Access*, 8:201071–201086, 2020.

[151] Adrian Hornsby and Rod Walsh. From instant messaging to cloud computing, an xmpp review. In *IEEE International Symposium on Consumer Electronics (ISCE 2010)*, pages 1–6. IEEE, 2010.

[152] Jorge E Luzuriaga, Miguel Perez, Pablo Boronat, Juan Carlos Cano, Carlos Calafate, and Pietro Manzoni. A comparative evaluation of amqp and mqtt protocols over unstable and mobile networks. In *2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*, pages 931–936. IEEE, 2015.

[153] Girija P Naik and A Umesh Bapat. A brief comparative analysis on application layer protocols of internet of things: Mqtt, coap, amqp and http. *International Journal of Computer Science and Mobile Computing*, 2020.

[154] Nguyen Quoc Uy and Vu Hoai Nam. A comparison of amqp and mqtt protocols for internet of things. In *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*, pages 292–297. IEEE, 2019.

[155] Zach Shelby, Klaus Hartke, and Carsten Bormann. The constrained application protocol (coap). 2014.

[156] Tanya Mohan Tukade and R Banakar. Data transfer protocols in iot—an overview. *Int. J. Pure Appl. Math*, 118(16):121–138, 2018.

[157] Manveer Joshi and Bikram Pal Kaur. Coap protocol for constrained networks. *International journal of wireless and microwave technologies*, 5(6):1–10, 2015.

[158] Andrew Yates, Kathryn Beal, Stephen Keenan, William McLaren, Miguel Pignatelli, Graham RS Ritchie, Magali Ruffier, Kieron Taylor, Alessandro Vullo, and Paul Flicek. The ensembl rest api: Ensembl data for any language. *Bioinformatics*, 31(1):143–145, 2015.

[159] Emanuele Viglianisi, Michael Dallago, and Mariano Ceccato. Resttestgen: Automated black-box testing of restful apis. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 142–152. IEEE, 2020.

[160] Ian Fette and Alexey Melnikov. The websocket protocol, 2011.

[161] Vanessa Wang, Frank Salim, and Peter Moskovits. The websocket protocol. In *The Definitive Guide to HTML5 WebSocket*, pages 33–60. Springer, 2013.

[162] Antonio J Jara, Alex C Olivieri, Yann Bocchi, Markus Jung, Wolfgang Kastner, and Antonio F Skarmeta. Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence. *International Journal of Web and Grid Services*, 10(2-3):244–272, 2014.

[163] Pascal Thubert and Jonathan Hui. Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. Technical Report 6282, RFC Editor, September 2011.

[164] Antonio J Jara, Pedro Moreno-Sanchez, Antonio F Skarmeta, Socrates Varakliotis, and Peter Kirstein. Ipv6 addressing proxy: Mapping native addressing from legacy technologies and devices to the internet of things (ipv6). *Sensors*, 13(5):6687–6712, 2013.

[165] Dominique Guinard, Vlad Trifa, and Erik Wilde. A resource oriented architecture for the web of things. In *2010 Internet of Things (IOT)*, pages 1–8. IEEE, 2010.

[166] Deze Zeng, Song Guo, and Zixue Cheng. The web of things: A survey. *J. Commun.*, 6(6):424–438, 2011.

[167] Dominique Guinard and Vlad Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences), Madrid, Spain*, volume 15, page 8, 2009.

[168] Matthias Kovatsch, Ryuichi Matsukura, Michael Lagally, Toru Kawaguchi, Kunihiko Toumura, and Kazuo Kajimoto. Web of things (WoT) architecture. `https://www.w3.org/TR/wot-architecture/`, April 2020.

[169] Sebastian Kaebisch, Takuki Kamiya, Michael McCool, Victor Charpenay, and Matthias Kovatsch. Web of things (WoT) thing description. `https://www.w3.org/TR/wot-thing-description/`, June 2020.

[170] Oscar Novo and Mario Di Francesco. Semantic interoperability in the iot: extending the web of things architecture. *ACM Transactions on Internet of Things*, 1(1):1–25, 2020.

[171] Zoltan Kis, Daniel Peintner, Cristiano Aguzzi, Johannes Hund, and Kazuaki Nimura. Web of things (WoT) scripting API. `https://www.w3.org/TR/wot-scripting-api/`, November 2020.

[172] Lee Feigenbaum, Gregory Todd Williams, Kendall Grant Clark, and Elias Torres. SPARQL 1.1 protocol. `https://www.w3.org/TR/sparql11-protocol/`, March 2013.

[173] Mike P Papazoglou. Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.*, pages 3–12. IEEE, 2003.

[174] Hugo Haas and Allen Brown. Web services glossary. *W3C Working Group Note (11 February 2004)*, 9:784–786, 2004.

[175] David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web services architecture, w3c working group note. *vol*, 2005:W3C, 2004.

[176] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[177] Zach Shelby, Klaus Hartke, and Carsten Bormann. The Constrained Application Protocol (CoAP). RFC 7252, June 2014.

[178] Amelie Gyrard, Pankesh Patel, Soumya Kanti Datta, and Muhammad Intizar Ali. Semantic web meets internet of things and web of things. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 917–920, 2017.

[179] Rabah Boucetti, Sofiane Mounine Hemam, and Ouassila Hioual. An approach based on genetic algorithms and neural networks for qos-aware iot services composition. *Journal of King Saud University-Computer and Information Sciences*, 2022.

[180] Alexandre Sawczuk da Silva, Hui Ma, Yi Mei, and Mengjie Zhang. A survey of evolutionary computation for web service composition: A technical perspective. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(4):538–554, 2020.

[181] Wuhui Chen and Incheon Paik. Toward better quality of service composition based on a global social service network. *IEEE Transactions on Parallel and Distributed Systems*, 26(5):1466–1476, 2014.

[182] Ying Chen, Jiwei Huang, Chuang Lin, and Xuemin Shen. Multi-objective service composition with qos dependencies. *IEEE Transactions on Cloud Computing*, 7(2):537–552, 2016.

[183] M. Essaid KHANOUCHE. *Sélection et composition de services sensibles à l'énergie et à la qualité de service en environnements ubiquitaires et intelligents ambiants*. PhD thesis, Université de A. Mira de Bejaia, 2016.

[184] Fabio Casati and Ming-Chien Shan. Event-based interaction management for composite e-services in eflow. *Information Systems Frontiers*, 4(1):19–31, 2002.

[185] Boualem Benatallah, Remco M Dijkman, Marlon Dumas, and Zakaria Maamar. Service-composition: concepts, techniques, tools and trends. In *Service-Oriented Software System Engineering: Challenges and Practices*, pages 48–67. IGI Global, 2005.

[186] Rania Khalaf and Frank Leymann. On web services aggregation. In *International Workshop on Technologies for E-Services*, pages 1–13. Springer, 2003.

[187] Ray Chen, Jia Guo, and Fenye Bao. Trust management for soa-based iot and its application to service composition. *IEEE Transactions on Services Computing*, 9(3):482–495, 2014.

[188] Demian Antony D'Mello, VS Ananthanarayana, and Supriya Salian. A review of dynamic web service composition techniques. In *International Conference on Computer Science and Information Technology*, pages 85–97. Springer, 2011.

[189] Ian Taylor, Matthew Shields, Ian Wang, and Roger Philp. Grid enabling applications using triana. In *Workshop on Grid Applications and Programming Tools*, 2003.

[190] Quan Z Sheng, Xiaoqiang Qiao, Athanasios V Vasilakos, Claudia Szabo, Scott Bourne, and Xiaofei Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218–238, 2014.

[191] Athman Bouguettaya, Quan Z Sheng, and Florian Daniel. *Web services foundations*. Springer, 2014.

[192] Chris Peltz. Web services orchestration and choreography. *Computer*, 36(10):46–52, 2003.

[193] Luca Mottola and Gian Pietro Picco. Programming wireless sensor networks: Fundamental concepts and state of the art. *ACM Computing Surveys (CSUR)*, 43(3):1–51, 2011.

[194] Qian Li, Runliang Dou, Fuzan Chen, and Guofang Nan. A qos-oriented web service composition approach based on multi-population genetic algorithm for internet of things. *International Journal of Computational Intelligence Systems*, 7(sup2):26–34, 2014.

[195] Zhen Yang and Deshi Li. Iot information service composition driven by user requirement. In *2014 IEEE 17th international conference on computational science and engineering*, pages 1509–1513. IEEE, 2014.

[196] Feng Gao, Edward Curry, Muhammad Intizar Ali, Sami Bhiri, and Alessandra Mileo. Qos-aware complex event service composition and optimization using genetic algorithms. In *International Conference on Service-Oriented Computing*, pages 386–393. Springer, 2014.

[197] Neeti Kashyap, A Charan Kumari, and Rita Chhikara. Multi-objective optimization using nsga ii for service composition in iot. *Procedia Computer Science*, 167:1928–1933, 2020.

[198] Jin Liu, Yuxi Chen, Xu Chen, Jianli Ding, Kaushik Roy Chowdhury, Qiping Hu, and Shenling Wang. A cooperative evolution for qos-driven iot service composition. *automatika*, 54(4):438–447, 2013.

[199] Neeti Kashyap, A Charan Kumari, and Rita Chhikara. Service composition in iot using genetic algorithm and particle swarm optimization. *Open Computer Science*, 10(1):56–64, 2020.

[200] Heba Kurdi, Fadwa Ezzat, Lina Altoaimy, Syed Hassan Ahmed, and Kamal Youcef-Toumi. Multicuckoo: Multi-cloud service composition using a cuckoo-inspired algorithm for the internet of things applications. *IEEE Access*, 6:56737–56749, 2018.

[201] Shangguang Wang, Ao Zhou, Mingzhe Yang, Lei Sun, Ching-Hsien Hsu, and Fangchun Yang. Service composition in cyber-physical-social systems. *IEEE Transactions on Emerging Topics in Computing*, 8(1):82–91, 2017.

[202] Mohamed Essaid Khanouche, Sihem Mouloudj, and Melissa Hammoum. Two-steps qos-aware services composition algorithm for internet of things. In *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, pages 1–6, 2019.

[203] Endong Tong, Lan Chen, and Huizi Li. Energy-aware service selection and adaptation in wireless sensor networks with qos guarantee. *IEEE Transactions on Services Computing*, 13(5):829–842, 2017.

[204] Zheng-yi Chai, Meng-meng Du, and Guo-zhi Song. A fast energy-centered and qos-aware service composition approach for internet of things. *Applied Soft Computing*, 100:106914, 2021.

[205] Mohamed Essaid Khanouche, Yacine Amirat, Abdelghani Chibani, Moussa Kerkar, and Ali Yachir. Energy-centered and qos-aware services selection for internet of things. *IEEE Transactions on Automation Science and Engineering*, 13(3):1256–1269, 2016.

[206] Osama Alsaryrah, Ibrahim Mashal, and Tein-Yaw Chung. Bi-objective optimization for energy aware internet of things service composition. *IEEE Access*, 6:26809–26819, 2018.

[207] Thar Baker, Muhammad Asim, Hissam Tawfik, Bandar Aldawsari, and Rajkumar Buyya. An energy-aware service composition algorithm for multiple cloud-based iot applications. *Journal of Network and Computer Applications*, 89:96–108, 2017.

[208] Gary White, Andrei Palade, and Siobhán Clarke. Qos prediction for reliable service composition in iot. In *International Conference on Service-Oriented Computing*, pages 149–160. Springer, 2017.

[209] Azadeh Gharineiat, Athman Bouguettaya, and Mohammed Nasser Ba-hutair. A deep reinforcement learning approach for composing moving iot services. *IEEE Transactions on Services Computing*, 2021.

[210] ZHOU Ming and MA Yan. Qos-aware computational method for iot composite service. *The Journal of China Universities of Posts and Telecommunications*, 20:35–39, 2013.

[211] Nguyen The Cuong, Federico Domínguez, Nguyen Thanh Long, Abdellah Touhafi, and Kris Steenhaut. Service composition with quality of service management in environmental sensor networks. *International Journal of Ad Hoc and Ubiquitous Computing*, 23(3-4):216–229, 2016.

[212] Liangzhao Zeng, Boualem Benatallah, Anne HH Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Transactions on software engineering*, 30(5):311–327, 2004.

[213] Sherry X Sun and Jing Zhao. A decomposition-based approach for service composition with global qos guarantees. *Information Sciences*, 199:138–153, 2012.

[214] Manisha Singh and Gaurav Baranwal. Quality of service (qos) in internet of things. In *2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU)*, pages 1–6. IEEE, 2018.

[215] Eyhab Al-Masri and Qusay H Mahmoud. Qos-based discovery and ranking of web services. In *2007 16th international conference on computer communications and networks*, pages 529–534. IEEE, 2007.

[216] Zhi-Zhong Liu, Dian-Hui Chu, Zong-Pu Jia, Ji-Quan Shen, and Lei Wang. Two-stage approach for reliable dynamic web service composition. *Knowledge-Based Systems*, 97:123–143, 2016.

[217] Andreia P Guerreiro, Carlos M Fonseca, and Luís Paquete. The hypervolume indicator: Problems and algorithms. *arXiv preprint arXiv:2005.00515*, 2020.

[218] Miqing Li and Xin Yao. Quality evaluation of solution sets in multiobjective optimisation: A survey. *ACM Computing Surveys (CSUR)*, 52(2):1–38, 2019.

[219] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, pages 1–11. IEEE, 2015.