République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université A.MIRA-BEJAIA

جامعة بجاية
Tasdawit n Bgayet
Université de Béjaïa

Faculté des Sciences Exactes
Département d'Informatique
Laboratoire d'Informatique Médicale LIMED

THÈSE

EN VUE DE L'OBTENTION DU DIPLOME DE DOCTORAT

Domaine : Mathématique et Informatique
Filière : Informatique
Spécialité : Réseaux et Systèmes Distribués

Présentée par
Monsieur EL SAKAAN Nadim

Thème

Automation mechanisms and large-scale applications of the Internet of Things and underlying technologies

Soutenue le 02/05/2024:

Devant le Jury composé de :

Nom et Prénom	Grade	Université	Rôle
Mr TARI Abdelkamel	**Professeur**	Univ de Béjaia	Président
Mr. AMROUN Kamal	**Professeur**	Univ. de Béjaia	Rapporteur
Mr. ELMIR Youssef	**MCA**	Ecole ESTIN, Béjaia	Examinateur
Mr. FARAH Zoubeyr	**MCA**	Univ. de Béjaia	Examinateur
Mr. ATMANI Mouloud	**MCA**	Univ. de Béjaia	Examinateur
Mr. SEBAA Abderrazak	**MCA**	Ecole ESTIN, Béjaia	Invité

Année Universitaire 2023/2024.

Acknowledgements and dedications

Above all, I express my gratitude to the benevolent Lord for the miracle of existence, the splendor of nature, the resilience and bravery bestowed upon me, the fervor for science and the virtue of patience instilled in me, and for the extraordinary family and supportive circle of friends He has gifted me.

I would like to extend my heartfelt appreciation to the most exceptional individual in the world, my mother, IZEGHLOUCHE Djamila. She has enriched my life and unwaveringly believed in me, making numerous sacrifices to bring me to this juncture. Her countless sleepless nights by my side, her selfless dedication to our upbringing, education, and well-being, and her willingness to give up so much for our betterment have left an indelible mark of gratitude in my heart.

Words elude me when it comes to expressing my indebtedness to my late father, Dr. RAGAB HASSEN Elsakaan. This humble endeavor serves as an opportunity to honor him, not only as an exceptional pediatrician but also as an unparalleled father. I extend my deepest gratitude to him for illuminating our path towards knowledge, science, and community service.

I would also like to thank my two brothers, professor RAGAB HASSAN Hani and doctor RAGAB HASSEN Ramy, for having opened the way for me, for having taught me so many things, for having been present at every moment in my life, for all the good times we spent together.

Without forgetting those who have provided guidance, assistance, and support during my research journey. I want to specially thank my supervisor the professor AMROUN Kamal for his patience, his presence and all the pleasant moments we have spent together over the years. I want to thank my co-supervisor the professor BOUABDALLAH Abdelmadjid, who has been by my side from the start of the scientific adventure, and who initiated and guided me through the scientific process.

I would also like to extend my appreciation to my life-long companion, Dr. BENNAI Soufia, who has stood by my side over the years and provided invaluable support. Furthermore, my heartfelt gratitude goes out to all those who have been a part of my life and have made their contributions, however small. To my family, friends, and neighbors, I offer my sincere thanks.

I wish to express my profound gratitude to an individual who has played a pivotal role in shaping my academic journey: my supervisor during both my bachelor's and master's degrees, Professor TARI Abdelkamel. He provided me with all the necessary resources and guidance to foster my growth as a researcher and educator. Initially, in the Department of Computer Science, then at the LIMED laboratory, and finally at ESTIN University. I would also like to take this opportunity to offer a special tribute to Professor BOUKERRAM Abdallah. He not only supervised the early stages of my doctoral thesis but also assumed a paternal role for the entire Faculty of Exact Sciences. His invaluable guidance, unwavering support, and remarkable human qualities have left an indelible mark. In memory of the late Professor ALOUI Abdelouhab, an exceptional educator, a humble and benevolent individual, whose teachings will forever resonate in our hearts.

I would like to express my sincere thanks to the entire staff of the Computer Science Department of the Faculty of Exact Sciences at the University of Bejaia, for the quality of their training, the social and scientific atmosphere, and the administrative responsiveness they have shown over the years.

I'd also like to thank and pay tribute to the masters of chaabi song, who kept me awake and accompanied me on my long scientific journey and sleepless nights. To Hadj Elanka, Amr Ezzahi and all the other masters of traditional Algerian song.

Contents

List of Figures

List of Tables

Acronyms

ACO	Ant Colony Optimization. 38
API	Application Programming Interface. 12
ASM	Active Status Message. 48, 53
BFT	Byzantine Fault Tolerance. 24
BW	Bandwidth. 34
CA	Certification Autority. 16, 83, 85, 86, 89, 91, 97, 102
CM	Candidacy Message. 48
CM	Cluster Manager. 63, 69
CPU	Central Processing Unit. 34
CRN	Cognitive Radio Network. 33
CSP	Cloud Service Provider. 19
DBFT	Delegated Byzantine Fault Tolerance. 24
DNS	Domain Name System. 13, 84
ECC	Elliptic-curve cryptography. 16
ECG	Election Concerned Gateways. 48, 51
EIM	Election Initialization Message. 48
FLF	Flooding Leaf Founding. 32
GLB	Global Load Balancer. 64
GPS	Global Positioning System. 31
GTS	Global Tasks Scheduler. 64, 70
GUI	Graphical User interface. 19
GW	Gateway. 48, 51–53
IaaS	Infrastructure as a Service. 12, 19
ICA	Imperialist Competitive Algorithm. 37
ID	Identifier. 30, 33, 83
IoT	Internet of Things. 3, 7, 12–15, 17, 26, 29, 30, 33, 34, 87, 102, 104, 105

LB Load Balancing. 20
LDNG List of Direct Neighbors: Gateways. 48
LDNN List of Direct Neighbors: Nodes. 48, 52
LLB Local Load Balancer. 64
LTS Local Tasks Scheduler. 64

MIPS Millions of Instructions Per Second. 60, 61

NDNG Number of Direct Neighbors: Gateways. 48
NDNN Number of Direct Neighbors: Nodes. 48, 51, 52

PaaS Platform as a Service. 12, 19
PBFT Practical Byzantine Fault Tolerance. 24
PCA Prodction/ Consumption Autority. 83, 86, 87, 90–92, 96, 102
PKI Public Key Infrastructure. 16
PoS Proof of Stake. 25
PoW Proof of Work. 24
PSO Particle Swarm Optimization. 36

QoS Quality of Service. 4, 39, 43

RAM Random Access Memory. 34, 60, 62
RF Radio Frequency. 12
RFID Radio Frequency Identification. 12
RH Requests Handler. 63
RPL Routing Protocol for Low-Power and Lossy Networks. 54
RSA Rivest–Shamir–Adleman. 16

SA Scoring Autority. 83, 86, 89, 90, 92, 96, 97, 102
SaaS Service as a Service. 12, 19
SEM Start Election Message. 48, 51
SHA Secure Hash Algorithm. 21
SLA Service-Level Agreement. 4, 34, 37, 43, 80, 81, 105

V2V Vehicle to Vehicle. 31
VANet Vehicular Ad-hoc Network. 29
VM Virtual Machine. 19, 20, 34, 36, 37
VTL Virtual traffic Lights. 29–32

WSN Wireless Sensor Network. 2, 12, 13, 26, 29–33, 47, 48, 57, 58, 105

Preface

The rise of the Internet of Things over the last few decades has transformed our relationship with the cybernetic world. It has blurred the boundaries between the realm and virtual world, making influences tangible in both directions. Through sensors capable of capturing information about the environment and actuators capable of acting on it, the boundary between the two worlds has been removed and this has given way to a whole new reality. All this has changed the scale at which system administrators must operate to provide functional safety and cyber-security services. We have gone from interconnecting small, local networks to a global network interconnecting billions of objects and users worldwide.

The set of models organizing the functioning of the Internet of Things into layers allows us to identify a trend: the presence of three main layers which are the perception layer, the core layer aggregating the network and service sub-layers, and finally an application layer. Our work focuses on identifying automation mechanisms at the perception and core layers, as well as on large-scale practical applications.

It is important to emphasize that a set of technologies forms the foundation on which the IoT infrastructure is built. Cloud Computing provides the computing power needed for processing data generated by connected objects, while Big Data offers analytical models to exploit these same data. Data can be obtained through sensors embedded in connected objects or collected by dedicated Wireless Sensor Networks (WSNs).

As the complexity of manual management and operation increases, the primary objective of this thesis is to equip operators with the necessary mechanisms to automate Internet of Things (IoT) operations to the greatest extent possible, thereby minimizing the need for human interventions. In navigating the landscape of IoT architectures, whether cloud-centric or edge-oriented, we assert that a cloud-based datacenter solution is the most pragmatic. Our focus is on developing solutions that enhance the automation of IoT processes within this organizational framework.

The thesis adheres to a conventional research procedure, commencing with an in-depth study of fundamental IoT concepts and an exhaustive review of recent and pertinent research endeavors. Subsequently, we introduce novel contributions positioned within the existing literature, backed by conclusive experiments. The overarching goal is to streamline IoT operations, reduce reliance on manual interventions, and foster a more efficient and automated paradigm for managing the complexities inherent in the Internet of Things.

During this thesis work, we conducted a comprehensive review of the state-of-the-art concerning automation across various levels. Our approach involved an in-depth exploration of the latest and most pertinent literature on the subject. Before formulating our scientific proposals, we meticulously examined and critically compared the most recent works in the field. This thorough review not only ensured that we were well-informed about the current advancements but also provided a solid foundation upon which to build our own contributions to the academic discourse. By synthesizing and critically analyzing existing literature, we positioned our research within the broader context of automation, allowing us to make informed and innovative contributions to the field.

In our research, we meticulously identified crucial automation mechanisms within each layer of the Internet of Things (IoT) architecture. At the perception layer, our emphasis lies on the election of a leader, a vital element for the self-organization of isolated Wireless Sensor Network (WSN). Shifting to the core layer, our focus extends to load balancing and task scheduling within a cloud computing environment. Lastly, within the application layer, our efforts are directed towards proposing a fully automated, large-scaled architecture that integrates the Internet of Things and blockchain technologies. This innovative approach aims to construct a reliable and privacy-aware ecological footprint calculator, addressing contemporary concerns in sustainability and environmental impact assessment.

Conclusive experiments were undertaken to showcase the effectiveness and significance of the proposed solutions. The obtained results, subject to detailed discussion, proved highly satisfactory, exhibiting notable performance and relevance. A comparative analysis with the most recent and pertinent related works further emphasized the robustness and efficacy of the proposed solutions.

This publication is a fulfillment of the scientific and ethical obligations mandated by research in the field of computer science. A collaborative effort among a group of researchers has contributed to its creation, encompassing the production of scientific articles and the subsequent results derived from the study. This work aligns with the standards and principles that govern scholarly pursuits, aiming to advance knowledge and contribute meaningfully to the scientific community.

Introduction

The Internet of Things (IoT) represents a technological paradigm that has transformed the interaction between human societies and cyberspace. Since its inception in the past decade, it has gradually and inconspicuously dismantled the barriers that once separated the physical realm from the digital world. Its ubiquity is evident as individuals remain constantly connected to smart objects, enabling access to information and other systems from anywhere at any time [1].

What sets it apart from previous iterations of the Internet is that it is no longer merely a straightforward support network with the sole task of transmitting data from one end to another. Presently, it can actively gather data from the environment through a perception layer, collate and aggregate this information, and subsequently translate it into actions that influence the surrounding world. Needless to say, the intricacy of such a network has given rise to various mechanisms designed, among other purposes, to automate a multitude of tasks associated with control and self-organization activities.

These distinctive features have allowed it to transcend the conventional use-case and seamlessly integrate into the realm of companies and organizations. Consequently, it has revolutionized information systems practices from what we have known until now. It has facilitated the automation of numerous laborious tasks, enhancing productivity and liberating human resources to concentrate on tasks with higher value-added [2].

Obviously, a network expanding at such astronomical pace must be designed to autonomously integrate new components, recover from failures, and make decisions in specific circumstances without human intervention. Our thesis work addresses this challenge by exploring ways to enhance the autonomy of the network at various levels. Recognizing the necessity for increased autonomy, we focused on interventions at the three operational levels of the Internet of Things (IoT): (i) At the perception level, our emphasis lies on leader election algorithms within the wireless sensor networks constituting the perception infrastructure of the Internet of Things. (ii) In the core and service layer, our efforts are directed towards implementing automatic load balancing and task scheduling among datacenter servers within a cloud environment. Finally, (iii) at the application layer, we figured out a massive deployment case of the IoT and underlying technologies to address a global problem which is climate warming [3].

In the era of the Internet of Things (IoT), the escalating proliferation of connected devices has emphasized the imperative for automation. The substantial number and heterogeneity of

IoT device manufacturers, coupled with the continuous demand for services from users, necessitate efficient and seamless automated processes. The complexity of managing and coordinating these interconnected devices, each with its specific functions and interactions, poses a formidable challenge. An automated approach is essential to streamline operations, enhance responsiveness, and reduce the need for human actions in handling tasks such as new devices joining or security imperatives. The need for automation in the IoT era arises from the desire to optimize resource utilization, alleviate operational complexities, and unleash the full potential of interconnected systems while liberating humans from redundant tasks to meet the growing demands of a digitally connected world.

Cloud computing is analogous to a nuclear reactor for the Internet of Things, providing immense computing power and facilitating centralization of user and connected object data. This shift has centralized security and recovery strategies in the era of the Internet of Things, fundamentally altering users' consumption patterns from ownership and investment to subscription-based models. At the core of this infrastructure, several collaborative mechanisms play a crucial role in maintaining the Quality of Service (QoS) and in meeting the Service-Level Agreement (SLA) that cloud service providers enter into agreements with their clients [4].

Blockchain is another technology that has utilized in our work. It functions as a data medium with a straightforward yet profoundly powerful structure. It allows transactions to be recorded in a decentralized ledger, where each block of transactions undergoes collective validation by numerous participants, and copies are distributed among them. This eliminates the necessity for a centralized data storage unit and guarantees the integrity of transaction data, along with non-repudiation by the involved users [5].

We initially introduced a novel algorithm for leader election in wireless sensor networks. This algorithm possesses the distinctive feature of being entirely distributed while including a fault tolerance module. In each area of the given network, a group of gateways competes to become the local leader. At the conclusion of this process, a leader is designated, and a list of potential substitutes is compiled to assume coordination in case of a failure. This algorithm provides a significant acceleration in terms of execution time as it omits the phase common to other algorithms, i.e., the construction of a spanning tree, and it also eliminates the need for rerunning in the event of a failure, thanks to the fault tolerance module [6].

Our second contribution centered on a novel algorithm that hybridizes tasks scheduling and load balancing in the cloud [7]. This algorithm possesses the unique feature of operating at two levels and in multiple stages, allowing a reduction in the complexity of the missions of these components and a robust decoupling of the roles of each of these modules. It utilizes k-means-based clustering to divide the datacenter into a set of clusters containing a bounded number of servers. Subsequently, it employs a round-robin method to assign a group of tasks to a specific cluster, before ultimately assigning them to the servers within that cluster based on a genetic algorithm. Consequently, load balancing concentrates on selecting overloaded clusters and servers that need relief from their workload, while the scheduling module is responsible for their rescheduling, thereby shortening the migration scheme typically done through approaches like ant colonies.

In the third contribution, we have introduced an innovative approach that, to the best of our knowledge, is unique. We addressed a significant issue, namely global warming, by proposing an infrastructure that governments should adopt to monitor ecological-impactful activities of the population and companies while respecting individual privacy [8]. This is facilitated by the ingenuity of the model we have modestly constructed, relying on three key components: (i) the Internet of Things for collecting behavioral data on consumption habits, moving away from declarative data that can be falsified; (ii) the blockchain to ensure anonymization, integrity, immutability, and, most importantly, non-repudiation of actions through the distributed ledger for storing the activities' footprints of companies and individuals; (iii) a public key infrastructure responsible for managing an asymmetric cryptographic system and an identification based on public keys. The advantage of this digital ecological impact calculator is that it does not require new technologies for implementation and ensures the accuracy of consumption data while respecting privacy.

We have crafted this thesis report with the intention of maintaining conciseness and minimizing the inclusion of literature elements to the essential amount necessary for comprehending our work. Simultaneously, we aimed to offer comprehensive details on our contributions, ensuring their clear understanding. The report adheres to a traditional three-phase structure:

1. The chapter 1 is dedicated to important theoretical concepts on technologies used in our work and can therefore be used to constitute background elements. Definitions and generalities on the Internet of Things can be found in section 2. Section 3 introduces the key elements of cloud computing, while section 4 is devoted entirely to blockchain.

2. The chapter 2 gives the state of the art and summarizes related works. Given the broad scope of the problem and that we have contributed at different layers of the Internet of Things model, it is a little bit more complex than it can commonly be found in other reports. Indeed, section 2 gives an overview on most important leader election algorithms, while section 3 reviews most recent and relevant approaches to ensure load balancing and tasks scheduling in the cloud. The last section is a summary of digital ecological footprint calculators.

3. We end this report by the chapter 3 in which we depict one by one our main contributions realized during this thesis. Section 2 presents a distributed and reliable leader election framework for wireless sensor networks. Then the section 3 introduces a novel hybrid multi-level load balancing and tasks scheduling algorithm for cloud computing environment. A proposition which is made for the first time as best as we know and is presented in section 4 to build a global privacy-aware infrastructure for a digital behavioural ecological footprint calculator.

4. A conclusion summarizing this report and giving some perspectives and future research directions is proposed in chapter 5.

Chapter 1

Background on computing key technological concepts in the Internet of Things era

1 Introduction

The main purpose of this chapter is to briefly present all the concepts required for the realization of the contributions we have made during this thesis and which are summarized in chapter 3. As explained in the introduction, our work focuses on automation mechanisms in cloud-based Internet of Things (IoT) environments on the one hand, and on fully automated IoT applications to achieve high value-added objectives on the other.

Hence, this chapter will introduce all key concepts and elements on technologies we used to build our contributions, it will dive deep on some critical points and give only an overview on some secondary others. The chapter is organized as follows:

First, the section 2 introduces the Internet of Things which is the cornerstone technology we rely on. The IoT is our main field of research and whether it is about integrated automation mechanisms or application automation, all the technologies required can be structured around and within the IoT. The three contributions we've made are set against the backdrop of the Internet of Things and related technological domains.

Then the section 3 is dedicated to cloud computing. The interest in cloud computing is twofold, firstly because the architectural organization of the IoT we are working on is cloud-centric, and secondly because in order to maintain a high level of performance, a certain quality of service and to respect service level agreements a lot of highly complex mechanisms collaborate together to automate and to streamline operations in cloud environments.

Finally the section 4 introduces the blockchain concepts. We end this chapter by introducing a last but not least important paradigm which is blockchain, or to focus on the aspect that interests us distributed ledger technology. Indeed, the blockchain offers a novel distributed, reliable and immutable support for data storage from which we draw a powerful tool to enhance non-repudiation of users actions in IoT environment.

2 Internet of Things

All required technologies for achieving our research work orbit around the Internet of Things (IoT), so it's easy to understand why we've chosen this as the entry topic in our background study. This section is dedicated to the study of the basic concepts of the IoT, its organizational models, the underlying technologies, the analysis of related risks and the various applications of this technological paradigm.

The Internet of Things represents a significant revolution in the field of computer science since its inception. Over the past decade, it has not only revolutionized the interaction between humans and cyberspace but has also influenced the approach of researchers to conventional problems. Consequently, this section will delve into the examination of concepts associated with the Internet of Things, emphasizing the aspects that have altered research practices.

We first give core concepts and main definitions in subsection 2.1, we then move to introduce architectural organizational models in 2.2 and underlying technologies in subsection 2.3. The subsection 2.4 depicts main threats and vulnerabilities on the one hand and presents how the IoT has transformed security services and public key infrastructure on the other one. To end this section some application fields are described within subsection 2.5.

2.1 Core concepts

The Internet of Things constitutes a ubiquitous network connecting billions of individuals and tens of billions of devices [9]. Diverging from the traditional internet, IoT establishes an interactive interface bridging the realms of cyberspace and the physical world. This is primarily due to the fact that connected objects are not merely computational devices but rather intelligent entities embedding sensors and actuators, enabling them to perceive and interact with the tangible world [10]. Notably, the prevalent trend of upgrading everyday objects into smart ones, simply by integrating a network interface card, central processing unit, and storage capacity, has accelerated the deployment of the Internet of Things. This rapid integration has addressed various requirements, such as utilizing global addressing methods and communicating via lightweight data exchange protocols

The Internet of Things can be characterized as a global network interconnecting diverse devices, comprising everyday objects enhanced with a network interface card, computational capabilities, and storage capacity [9]. This extension of the conventional internet possesses the capability to capture environmental events and interact with the surroundings, enabling a myriad of applications across domains such as home automation, industry, and agriculture. Our interest in IoT for this work stems from two key aspects: (i) the availability of diverse architectures that integrate various technologies, including cloud or fog computing and massive data management systems leveraging big data. This enables the utilization of computing capacities across components and scales. (ii) Its ubiquity—from cloud platforms to homes and personal areas through personal area networks—making it omnipresent and capable of performing diverse tasks without the need for additional hardware [11].

It is important to note that the Internet of Things (IoT) encounters numerous security challenges. The interconnection of various technologies within IoT introduces a multitude of underlying threats. To gain widespread acceptance and successfully deploy IoT comprehensively, traditional security services must be ensured, and new mechanisms should be employed to establish a certain level of trust between users and IoT. Unfortunately, classical security approaches are not applicable in this context due to the nature of the devices (connected objects) constituting the network. These devices have processing units and memories with very limited capacities, making it impractical to implement standard protocols [12]. Public key infrastructure, coupled with digital certificates, represents one of the most effective ways to manage identities and distribute cryptographic keys in large networks. Consequently, many researchers have endeavored to create convenient and lightweight versions that maintain the same security level while reducing computing and networking overhead [13]. We will leverage these propositions to handle the identification and distribution of public keys in our contribution system 4 through trusted certification authorities.

2.1.1 Definition of smart objects

Defining an object is not an easy task; however, we operate under the assumption that it can be considered as an object or a smart thing any entity with a unique global identifier, possessing a physical existence and a collection of associated physical features. Furthermore, it should possess minimal communication abilities and basic computing capabilities. Eventually, an object can sense and/or interact with the real world [14].

For an object to be part of the IoT ecosystem, it must fulfill specific minimal prerequisites: (i) It should be equipped with a network interface card featuring a distinctive global identifier, (ii) It must integrate a central processing unit capable of executing dedicated cryptographic and lightweight data processing protocols, and (iii) It needs to have a sufficiently large memory capacity to store security parameters and firmware data [15].

Since objects are manufactured and have a service life before obsolescence, they adhere to a specific life-cycle describing all steps since their production until their revocation [16]:

- Manufacturing: the object is created and physical features are integrated to it (Physical address, set of embedded devices).
- Bootstrapping: the object is integrated or deployed to its functional environment, configured (network address attribution, role acquisition, setting different parameters) and started (work beginning on the network).
- Operational phase: the object works conforming to his initial specification and evolves in the network with respect to comportment rules.
- Maintenance: if the object fails, a physical or software reparation can be invoked by an automatic failure detection system. The object is repaired and then restarted from the bootstrapping phase.
- Revocation: The last step of an object life-cycle is the revocation when it is definitely out of service or considered as old made.

2.1.2 System-level characteristics

The Internet of Things unlike its predecessor presents some interesting properties due to the nature of the linked things and the used intelligent algorithms, we can summarize them here [9]:

- Heterogeneity: each device may exhibit a unique set of capabilities and be constructed by integrating components from various vendors. The IoT addresses methodologies to enable seamless communication and collaboration among these objects in a fully automated and transparent manner.
- Dynamism and spontaneous interaction: this network is distinguished by its high mobility, where objects can dynamically join the global network or a specific sub-network and depart at any time. As a result, new links are constantly formed, and old ones are severed, leading to an undetermined and evolving topology over time.

- Resource constraints: the majority of connected devices are battery-powered, transforming management systems as a significant portion of the network is transient and needs to incorporate charging phases in its life cycle. Energy is not the sole constrained resource, but it serves as a bottleneck, requiring optimization in the consumption of communication and processing capacities. To address these limitations, researchers have introduced an entirely new ecosystem of algorithms and protocols that prioritize energy efficiency.

2.1.3 Common layered-model

To elucidate the operational principles of the Internet of Things and the associated technologies, researchers and international organizations have introduced various layered models illustrating the components of IoT, from the application perspective to the physical elements and connections [15]. Figure 1.1 depicts a widely accepted stack of layers [17].

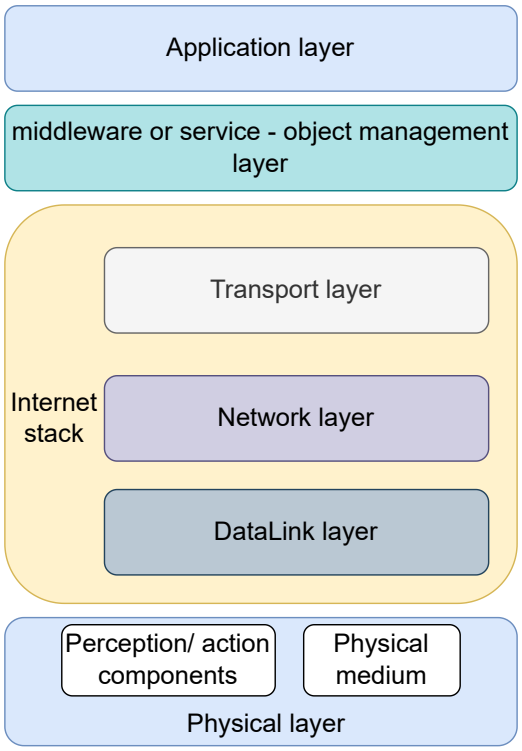

Figure 1.1: IoT layered model

Each layer provides services ensured by a set of standard protocols to higher layer, this model can be summarized as follows:

- Application layer: this layer creates the interaction interface with end users, essentially enabling the exposure of high-level applications and data exchange protocols.
- Middleware layer: this layer provides an abstraction, facilitating the development and the use of high-level applications without requiring in-depth knowledge of physical or network components. It offers the upper layer a set of virtualized functions in a modular way,

making the lower layers transparent, and delivers services through simple access points [18].

- Internet stack: the intermediate layers involved in ensuring network services vary between architectures. To avoid a lengthy dichotomy, we focus on the most commonly used among them. This level plays a crucial role in interconnecting disparate objects, connecting them to cloud platforms, and ensuring a certain level of quality in data transport [9].
- Physical layer: This layer regroups two kinds of components:
  - Devices: which can be sensors capturing environment parameters, actuators realizing basic actions in their immediate environment, or more complicated objects like smart things [15].
  - Physical medium: regrouping physical connection technologies like bluetooth, wifi and so on [9].

2.2 Organizational models

The Internet of Things is engendered around a hybridization of two previous paradigms, a Cloud based computation on one hand, and a fully decentralized and distributed computing on the other hand [12] [19].

2.2.1 Cloud-centric model

Cloud computing has emerged as a viable alternative to on-premise infrastructure and personal computing appliances in the last two decades. It provides tailored services over the Internet, spanning from hardware to applications. Endpoint devices in the IoT are capacity-limited. Therefore, a Cloud-centered architecture seems to be the only reliable alternative. In this type of structure, connected objects primarily collect data, transfer them to Cloud servers, perform some light tasks locally, or make actions in the environment. The major part of the processing occurs on the Cloud side [20].

2.2.2 Edge and fog oriented model

Another approach emphasizes what is known as edge computing. In this architectural arrangement, researchers highlight the potential of end-objects when combined. Individually, smart things exhibit mediocre performance, but their collaboration yields substantial computational capabilities. It is obvious that not all problems can be distributed, and access to resources is restricted by control policies; therefore, this solution has limitations when viewed from this perspective [11]. Building on this concept, researchers propose deploying devices with relatively high performance throughout the network to achieve an optimized number and distance to end devices. This strategy aims to reduce response times and network saturation compared to the Cloud-based approach. Furthermore, with fog computing, other specialized devices are employed to bridge the gap between edge computing devices and Cloud servers [21].

2.3 Underlying technologies

Internet of Things exploits and integrates a lot of existing technologies in order to make a global automatic system, between these technologies, main ones are cited here [22]:

- Radio Frequency Identification (RFID): is one of the most extended identification systems, these systems comprise (i) Radio Frequency (RF) Tags; a tag has some computation and storage capabilities, and (ii) RF readers or transceivers. RFID technology is lower cost than previous solutions (barcodes: UPC and EAN), and it's more secure, the penetration of RFID systems is mainly limited by privacy concerns [23].

- WSN: is an elementary component of IoT, one of the fundamental characteristics of automated networks is the ability to sense its environment, this is made possible thanks to WSN. WSN play an important role in connecting physical and virtual world together [22]. wireless sensor networks (WSN) Wireless Sensor Networks (WSNs) represent a significant technology in the digital revolution [24]. They serve as the equivalent of senses for computer and object systems, constituting the perception layer in a typical layered model. A standard hierarchical representation of WSN breaks them down into two levels: (i) Gateways, responsible for collecting data from sensors and relaying it to base stations, and (ii) Sensors, tasked with capturing environmental information [25] [26].

- Cloud Computing: provides multiple services in three principle forms on the IoT; (i) Infrastructure as a Service (IaaS) this kind of service provides users storage and computing resources. (ii) Platform as a Service (PaaS) allow users to develop and deploy application on the cloud using tools and Application Programming Interface (API). (iii) Service as a Service (SaaS) provides applications for daily uses or interfaces to these applications [27].

- Big Data: the huge size of data generated by open networks, low cost storage and data sharing, automation of devices treatments and synchronization, leads a lot of searchers to find numerous techniques and politics for data management and coherence keeping. these approaches are more adapted for IoT context than classical data-base algorithms and methods. Indeed, regarding limited storage capacity of some kinds of thing, it's easier, for edge nodes, to analyze data flows than to filter and store information [28].

- Blockchain: the blockchain is one of the major revolutionary technologies developed in the last decade. Initially transforming the way we handle transactions, it has subsequently proven to have various applications in fields such as the IoT security.

2.4 Security analysis

This subsection gives an overview of the IoT security, Starting with the study of vulnerabilities, up to the analysis of security products dedicated to the IoT. In particular, we'll focus on the security services identifying which ones are more relevant depending of the context.

2.4.1 A natural vulnerable network

The Internet of Things presents a lot of vulnerabilities due to its nature, some searchers describe it like: "interconnection of threats". An example of this threat is the attack on 21

October 2016. This attack was made possible by the large number of unsecured connected objects over the internet global network, by unsecured we mean devices using default-passwords, ignorance of users who download and install applications from unknown sources and without verifying required permissions and so on. The attack directs a huge amount of bogus traffic at the victim servers (Dyn Servers: Domain Name System (DNS) services company).

The IoT encounters numerous security challenges. For instance, in recent years, there has been a rise in the number of distributed denial of service attacks based on botnets targeting smart devices [29]. As illustrated in Figure 1.2, attacks on the IoT can be categorized based on the targeted layer, as outlined in [11].

- Hardware or physical attacks: given the characteristics of the IoT, certain devices may be deployed in remote and unmonitored areas, creating an opportunity for attackers to physically access and manipulate the device's nature or functionality. Once in possession, an attacker could potentially harm the device, attempt to brute-force access to its cryptographic keys, or modify its firmware to execute actions on their behalf within the network.

- Network attacks: an attacker may exploit the network stack, engaging in activities such as conducting a man-in-the-middle attack by poisoning network services, altering a device's behavior to turn it into a sinkhole appliance, or simply sniffing traffic to gather data for potential future attacks.

- Application or software attacks: applications are not exempt from threats. At this level, an attacker can exploit weaknesses resulting from poor programming processes to inject malicious scripts, or take advantage of users' lack of awareness about best practices, thereby conducting social engineering-based attacks.

2.4.2 Threats and vulnerabilities

The Internet of Things is an interconnection of existing technologies which presents numerous vulnerabilities, when exposed on a global network these vulnerabilities are scope of malicious people and become threats. Indeed, attackers can exploit any present security breach to damage the system. Internet of Things is composed of objects carrying limited storage and computing abilities. This fact make impossible the use of classical security mechanisms letting appear numerous threats that can be exploit by an attacker in order to harm the victim system's. Among threats, some are more harmful, we summarize them here [30]

- Weak encryption: due to the low computing capacity of the objects, cryptograms have to be light, it results a low level of data protection.

- Low-energy level: most of things are powered by a battery, a DoS attack consists to solicit a sensor node, for example, until it consumes the entire battery.

- Compromising a sensor: A second effect of auto-integration, is the reducing of access control level, it results that a compromised node has more chance to be accepted in a network than in a classical WSN.

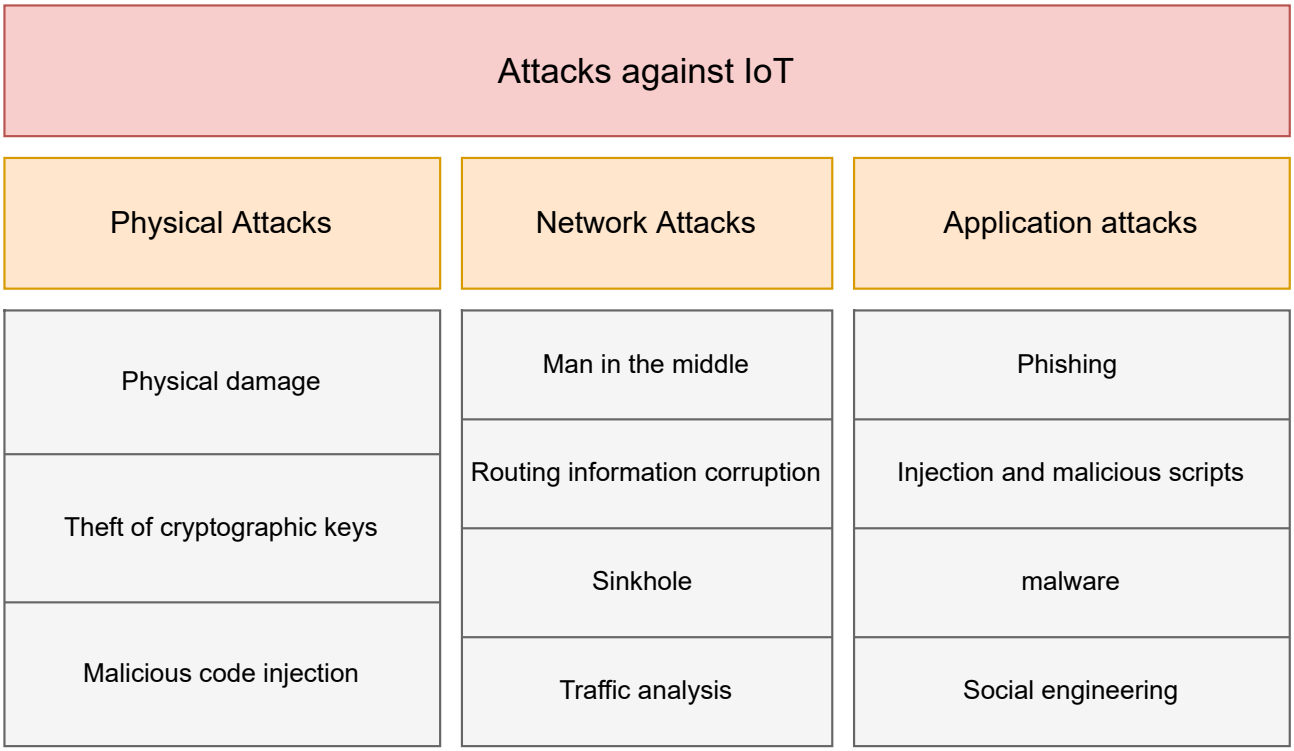

Figure 1.2: Common attacks against IoT

- Infringement of privacy: wearable things and monitoring smart spaces makes violation of privacy by: localization, tracking, filming, alerting, and so on. Guidelines were proposed to ensure privacy in this kind of environment, the main approach is to notify and make the user accepting conditions.
- Undesirable actions: actuators make IoT able to impact the real world, hacking an actuator can result in an undesirable action.
- Open data: IoT adopts the principle of open data, so any information exchanged on the network can be accessed by authorized users. Getting access to a specific domain grants the access to relative data flows, here again, auto-integration can offer opportunity for attackers.
- Admissibility and corrupted data creation: we have to make attention to each node we integrate into a network and if its data flow can be trusted or not, corrupted data can lead to make wrong decisions and harm the environment.
- Identification: one of the most crucial aspect in IoT is the identification. So we need to have mechanisms ensuring protection against identity usurpation and modification of identification data.
- Physical Access: As the nodes are arbitrarily deployed in a large and open environment. Attackers can directly and physically try to access the debug port of the object in order to control it.

- Rogue Access Point: this is an access point which doesn't differ from normal one, but it is used in order to eavesdrop the communication going through it.

2.4.3 Security services

In order to ensure user confidence and sustainability, researchers have enhanced classical approaches to provide the required security services. These challenges can be reviewed by categorizing them under the corresponding service [31].

- Authentication and Access Control: historically, authentication sought to offer ample evidence for confirming an assumed identity. Within the realm of the IoT, authentication functions across different stages: (i) user authentication, involving the system's scrutiny of credentials or biometric parameters via an end device such as a smartphone; (ii) device authentication, a pivotal stage ensuring the ownership and identifier of a device before establishing a connection; and finally, (iii) message authentication, incorporating information into each message to facilitate the verification of the sender's identity.

- Privacy: addressing privacy issues within the IoT entails broadening policies to manage user-controlled data exchange and the continuous capture of data without the user's awareness. Merely encrypting messages is no longer adequate, given that data from smart devices in the user's environment are consistently captured and transmitted, resulting in permanent identification and ongoing tracking.

- Integrity: verifying the integrity of information or resources assumes a distinct dimension within the IoT. A multitude of novel approaches leverage data from nearby devices or depend on artificial intelligence algorithms to identify alterations or corruption in the data.

- Availability: ensuring the availability of services in the Internet of Things (IoT) poses a persistent challenge, particularly due to a substantial portion of network nodes relying on battery power. Conventional methods are bolstered by mechanisms that facilitate an automatic transition to sleep mode during standby phases.

2.4.4 Public key infrastructure

Asymmetric or public-key cryptography refers to cryptographic systems in which the keys employed for encryption and decryption operations are not identical but exhibit a mathematical complementary property [32]. The concept revolves around each node having two keys: a public key, accessible to every entity on the network, used for encrypting messages destined for the node, and a complementary private key, known only to the node, employed for message decryption. This category of algorithms offers robust resistance to attacks and has gained prominence in traditional networks due to the simplicity of key sharing and the non-propagation of damage when a node is compromised. Unfortunately, the drawback of these approaches is their resource-intensive implementation, making them impractical in the context of the Internet of Things [33].

One of the most frequently utilized asymmetric cryptography algorithms is Rivest–Shamir–Adleman (RSA), employing keys with a length of up to 4096 bits. However, RSA is confronted with three primary limitations: (i) it necessitates a large number of keys with significant sizes to ensure a high level of security, (ii) it is slower and more intricate to implement than equivalent symmetric algorithms, and (iii) it is susceptible to indirect attacks such as known or chosen plaintext, among others [34]. Given the computational constraints of smart devices, RSA becomes impractical, prompting the need for lightweight approaches. One notable alternative is Elliptic-curve cryptography (ECC) methods [35] [36]. In a comparative study by Vidya et al [13], it is demonstrated that a crypto-system based on ECC offers a higher level of security with key sizes ranging from 160 to 256, as opposed to 1024 to 4096 for RSA, all while reducing the time required for key generation and encryption/decryption operations.

The digital signature employs the principle of asymmetric cryptography to establish a proof of integrity for resources. A common use case involves verifying the integrity of a message. In this scenario, when a sender sends a message to an entity, the sender hashes the message and encrypts the resulting data with its private key. Upon receiving the message, the recipient entity decrypts this signature using the sender's public key. Subsequently, the recipient hashes the entire message and compares it to the result of the previous operation. If the two match, it indicates that the message has not been altered [37].

Ensuring the reliability of key sharing poses a significant challenge in the widespread use of asymmetric cryptography. The bootstrap step, which involves the initial key exchange, creates vulnerabilities and opens the door to various threats. To address this issue, digital certificates play a crucial role. These certificates serve as numerical identity cards containing information about the holder, such as a global identifier and public key. The integrity of these certificates is guaranteed by trusted servers or authorities on the network, who sign the certificates with their private keys. This process highlights the importance of digital signatures in securing the delivery of certificates [38].

Certificates serve different purposes and can be categorized as either identity certificates or attribute certificates. In the case of identity certificates, they convey identification details along with public key information. On the other hand, attribute certificates contain specific pieces of information that grant access to a predetermined set of services or resources [39].

Management of these certificates involves essential architectural components, particularly cloud-based servers trusted by all entities building what is called Public Key Infrastructure (PKI) [40]. These components provide the following key functionalities:

- Certificate Creation: the Certification Autority (CA) should issue a certificate for each user generating a new public key or modifying an existing one.
- Distribution and Verification: entities aiming to verify the validity of a certificate for a specific node in the network, before initiating a communication session, can achieve this goal by interacting with the trusted authority that issued the certificate.
- Revocation: In the event of a key compromise or if a node changes its key for security reasons, the CA must promptly revoke the corresponding certificate.

2.5 Applications

The Internet of Things as said is omnipresent and offers a lot of opportunities in a very large and various domains of application, for example we can find its usage in following fields [20] [22]:

- Smart homes and cities: provides functionalities including monitoring structural integrity and environmental parameters, automating home control, optimizing delivery times for various services, and more.
- Industry, retail and logistics: the IoT facilitates product tracking and organization, streamlines fleet routing processes, and assists in managing assets and inventory information.
- Healthcare: the Internet of Things (IoT) enables the creation of reactive, mobile environments for monitoring the health and activity levels of individuals, providing, among other things, improved medical follow-up and reduced latency in emergency care.
- agriculture and environment: Massive, industrialized agriculture has been taking over for decades to meet the growing needs of populations and ensure human food security. The Internet of Things (IoT) now makes it possible to monitor growth, optimize the use of aquatic resources and enhance manage major risks.
- E-commerce and trading: even though this segment is the one that has cost connected objects the most criticism, the IoT has enabled improved targeting of advertising campaigns thanks to the behavioral data collected and profiling operations assisted by machine and deep learning.

Table 1.1 gives an overview of application fields with main mechanisms ensured or automated by the Internet of Things. It is obvious that this list cannot be exhaustive.

2.6 Conclusion

This section allowed us to review the important concepts involved in the Internet of Things: basic definitions, components, architectures, underlying technologies, security aspects and applications. These concepts are cornerstone in the realization of our contributions and will allow a better understanding.

The next section will introduce cloud computing. As previously said we focus on a cloud-centric architecture of Internet of Things and it will be hard to understand how it works on the one hand and why our contribution described in section 3 is important without keeping in mind some key concepts on cloud computing.

3 Cloud computing

Given that we're working on a cloud-based version of the Internet of Things, it's really important to understand the ins and the outs of this technology and its functional model. Having briefly discussed the cloud-centric architecture of the Internet of Things in the subsection 2.2

Field	Mechanisms
Smart Buildings	Air conditioning Detection systems Smart appliances
Smart cities	Intelligent roads VANETs Emergency systems Smart health structures Smart schools
Healthcare	Health and fitness monitoring Wearable Electronics
Agriculture and environment	Green house control Monitoring growth of plants Smart irrigation Weather monitoring Air and noise pollution monitoring Fire detection systems
Industry, retail and logistics	Machine diagnosis Inventory management Route generation and fleet tracking industrial conditions monitoring.
e-commerce	Smart payments Inventory management Smart suggestions and advertising

Table 1.1: IoT Application fields

from the previous section, we're going here to take a closer look at the cloud-related elements we consider important for our work.

In broad terms, cloud computing denotes the provision of hardware and software services directly over the internet. The nature of these services depends on the specific requirements of the customer and is governed by particular constraints outlined in the agreement contract between the cloud service provider and the end-user. This section discusses the key elements involved in cloud computing.

This section gives some preliminary definitions in subsection 3.1, then it depicts the commonly used architectures and models of service provisioning respectively within subsections 3.2 and 3.3. The section ends with subsection 3.4 presenting the main mechanisms collaborating to ensure quality of service and service-level agreement in cloud environment.

3.1 Preliminary elements

Cloud computing emerged as a transformative technology almost a decade ago, introducing a

revolutionary paradigm by delivering resources as services directly over the internet to individuals and companies. These resources can encompass hardware, software, or take various other forms, presenting advantages such as elasticity, a pay-as-you-go model, multi-tenancy, and more [41].

3.2 Architectural model

Upon examining various potential architectures and organizational structures for cloud operations, we have determined that, irrespective of the model, the architectural elements can be categorized into one of the three levels depicted in Figure 1.3:

- Requests handler: encompassing a set of components responsible for collecting requests from clients and retrieving pertinent information, such as task length, priority, deadline, required data, and so forth.
- Data-center controller: assumes an orchestration role by receiving information from the requests handler regarding tasks to schedule and being provisioned with available resources by the resource manager. Subsequently, the controller employs a scheduling strategy to determine the assignment of tasks to specific Virtual Machine (VM) and the allocation of those VM to particular hosts.
- Resources manager: responsible for monitoring the states and utilization rates of hosts and virtual machines, it supplies vital information on which the controller relies for task scheduling and load balancing.

3.3 Service provisioning models

The cloud providers deliver services in several standardized manners among which we retain [42]:

- Infrastructure-as-a-Service (IaaS): the hardware is delivered to the client, who is responsible for installing all stack components on the hardware, including operating systems, middlewares, runtime environments, and applications. The cloud provider is solely responsible for managing the hardware portion.
- Platform-as-a-Service(PaaS): the supplier's responsibility is elevated further up the stack, as they will be responsible for installing and managing operating systems, middlewares, and all required execution environments.
- Software-as-a-Service(SaaS): in this pattern, the client interacts with cloud services through a Graphical User interface (GUI). All necessary services are delivered as ready-to-use applications, and the provider assumes responsibility for the entire infrastructure stack.

3.4 Operational mechanisms

Regardless of the specific features of a cloud service provider (Cloud Service Provider (CSP)), they all need to incorporate a minimum set of mechanisms that collaborate in order to ensure the smooth running of the environment and continuity of service delivery. Among these mechanisms, we will introduce here the most important ones that are relevant to our work.

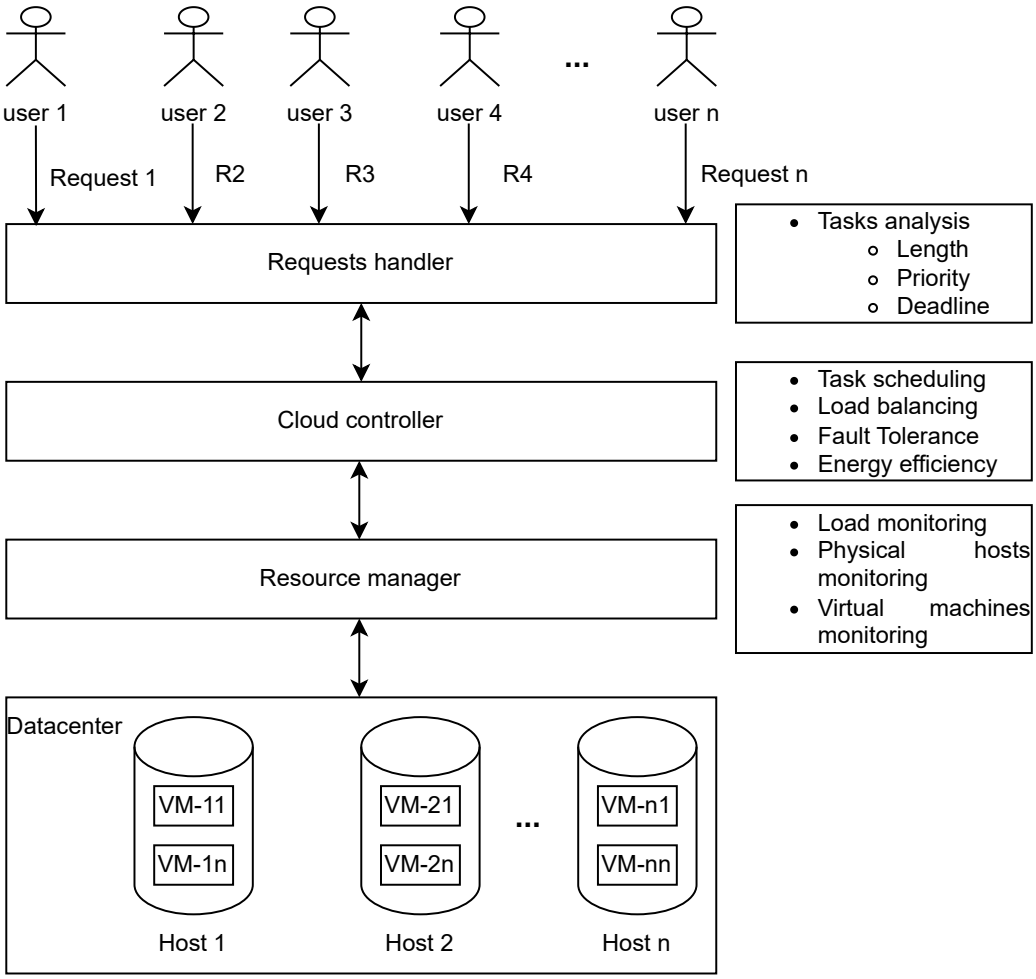

Figure 1.3: Cloud datacenter organizational architecture

3.4.1 Tasks scheduling

This mechanism is responsible for task allocation across servers. Factoring in a set of constraints, it determines the optimal resource for a given set of tasks. This can be achieved through static or dynamic means, depending on whether the scheduler relies on prior information about tasks and resources for decision-making or continuously monitors node behavior to determine the most suitable node for a specific job. The scheduler can be preemptive, allowing tasks to be interrupted during runtime, or non-preemptive. Additionally, it can operate in an online or offline manner based on whether tasks are directly planned on resources or grouped in batches beforehand [43].

3.4.2 Load balancing

Load Balancing (LB) plays a pivotal role in ensuring optimal functionality in a cloud environment. Its purpose is to delineate techniques responsible for distributing the workload across servers within a datacenter. Essentially, LB is the methodology employed to sustain equilibrium in the resource utilization of servers, preventing overloading or underloading. Load balancing can be executed at one of two levels: (i) virtual machines (VM) level or (ii) hosts level. In the

former case, the load balancing algorithm deals with the workload on virtual machines, overseeing the distribution and migration of tasks to maintain favorable workload partition conditions. In the latter case, it manages the distribution of virtual machines across physical servers.

3.4.3 Fault tolerance

Fault tolerance assesses a system's ability to recover after a failure occurs, where a failure is defined as a sequence of undesirable actions leading the system to an unsuitable or non-specifications-conforming state. Two primary families of fault tolerance approaches can be identified: (i) proactive, wherein technical efforts focus on anticipating failures and minimizing their impact on the overall system, and (ii) reactive, which concentrates on methods to swiftly recover after a fault occurs and restore the system to its last known coherent state. Various established approaches, such as hardware redundancy, job replication, checkpoint and restart, and others, are employed to ensure reliability, availability, and integrity in the cloud environment [44].

3.5 Conclusion

This section introduced cloud computing paradigm and has given major components, organizational models and service provisioning patterns. All these elements are crucial if we are to approach the automation mechanisms we're interested in, namely load balancing and tasks scheduling. We have been able to present them in general terms in this section, and we will require this to understand the contribution we made in section 3.

The next section will be dedicated to the blockchain. It is the last required building-block for our work. The section will give an overview on preliminary elements and focus on consensus protocols which is the most automated level in this storage technology.

4 Blockchain

The third important technological paradigm we used is blockchain. It refers to a storage model which is fully distributed and immutable. Instead of a central server maintaining the entire database like in classic relational models, in a distributed ledger each user called participant is asked to validate a number of transactions and to keep locally a partial version of the final blockchain.

Blockchain stands as a foundational technology in our contemporary era. It operates as a distributed ledger, taking the form of a database composed of blocks. Each block contains a header with integrity check information and a set of transactions, including the Secure Hash Algorithm (SHA)-3 hash of the previous block and the hash of the current one. The ledger is maintained in a fully distributed manner by nodes (workers or miners). In other words, each node in the network possesses a partial or complete copy of the chain [45]. The reliability of this participatory storage method has been exemplified by its use in cryptocurrencies, demonstrating high performance in preserving data integrity without compromising participant identities

[46]. Beyond finance, this principle has found applications in various domains such as health, cybersecurity, and supply chain management [47].

This section is dedicated to blockchain concepts, it starts by some key concepts in subsection 4.1, it continues by presenting the working manner and most commonly used consensus protocols in subsections 4.2 and 4.3. Finally it ends by subsection 4.4 giving some use cases.

4.1 Definition and structure

The blockchain is characterized as a fully distributed ledger composed of blocks, each containing a specific number of transactions and cryptographic links to one another, as illustrated in Figure 1.4. Moreover, its appeal in various fields stems from key elements, as outlined in [48]:

- Decentralized: it operates in a fully distributed manner and does not rely on central servers.
- Immutable: deleting any information from the blockchain is prohibited. To reverse a transaction, a new one with reverted data must be inserted into a new block.
- Privacy preserving: it does not necessitate any identification information; mechanisms for ensuring trust between nodes are inherently built within it. It's crucial to emphasize that the recorded data are visible to every node in the network.

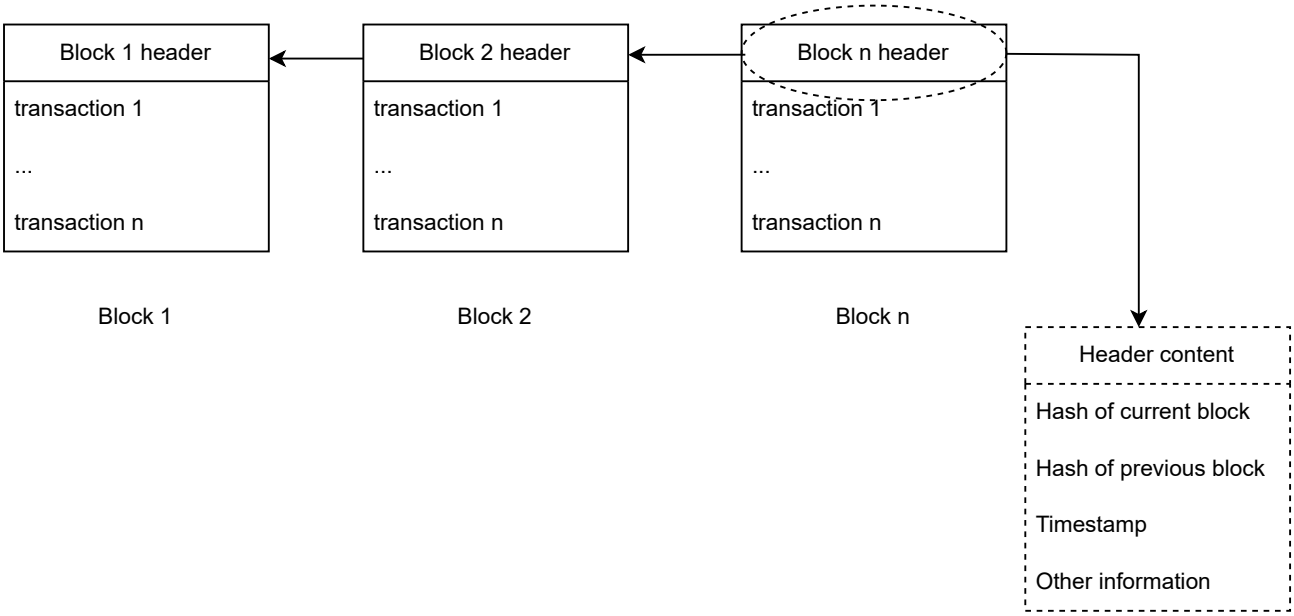

Figure 1.4: Blockchain architecture

4.2 Layered model and typology

An alternative approach to enhance the utilization of blockchain involves adopting a layered model. Although it doesn't have a fixed structure, it can be represented in the stack depicted in

Figure 1.1. The application layer facilitates user interaction and specific application programming through dedicated interfaces. The core of the blockchain is situated in the operational layer, which is further divided into sub-layers responsible for (i) smart contracts describing transaction processes, (ii) consensus protocols used by nodes to validate block additions, and (iii) network elements managing nodes participating in the blockchain. The data layer comprises pure data blocks containing the included transactions [47].

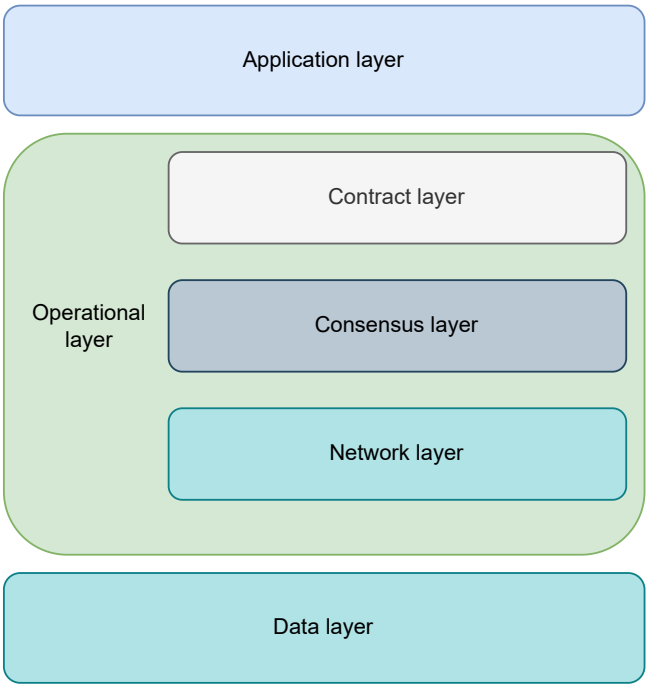

Figure 1.5: Blockchain layered model

According to the accessing and participating mode there are several types of blockchain [46] [49]:

- Public: in this type of blockchain, any node can join the network and possesses complete rights to engage in transactions and consensus protocols.
- private: in this scenario, the blockchain adheres to an access control and privileges management policy. Therefore, nodes must undergo authentication before joining and are restricted to perform only authorized actions [50].

4.3 Consensus protocols

To uphold the coherence of the blockchain, before adding a new block, the involved nodes must verify the integrity of transactions and filter out invalid ones. Additionally, they need to reach a consensus on the order of appending these new blocks to the existing chain. The steps involved in verifying transactions and ordering blocks depend on the chosen consensus protocol [47].

The choice of consensus protocol is evidently influenced by the specific use case and context. We can succinctly categorize the most common protocols based on whether they are used in permissioned or permissionless blockchains [46]:

4.3.1 In permissioned blockchain

Depending on the kind of network type we can retain two permissioned blockchain environments with specific consensus protocols:

1. Synchronous network: in these networks, communication operates under constrained delays and adheres to a common time clock. Therefore, consensus protocols need to incorporate these constraints into their operating mode. Examples include [51]:
    - Byzantine Fault Tolerance (BFT): this category outlines a general approach to tolerate up to one-third of corrupted nodes in a network while still maintaining correct functionality. It's noteworthy that achieving consensus in the system requires validation by at least two-thirds of the participants.
    - RAFT: this approach relies on two key concepts: (i) consensus is achieved when N/2 + 1 nodes agree on it, and (ii) the network is divided into clusters, each with a unique leader responsible for exchanging information with external elements and informing its followers.
2. Asynchronous network: in such systems, there are neither limited delays nor common clocks, and the protocols used must adapt to ensure a reasonable quality of service and response time. Notable protocols in this category include [46]:
    - Practical Byzantine Fault Tolerance (PBFT): it is an enhancement of the BFT approach designed to make it practically usable. In this approach, consensus is achieved by receiving favorable responses from the number of maximum tolerated faulty nodes plus one. Similar to the BFT approach, primary nodes are elected to manage tasks and communication, while secondary nodes act as miners. When a request is received by a leader, it is forwarded to workers who process and respond to the requester. The transaction is validated when the specified number of nodes provide the same approval response.
    - Delegated Byzantine Fault Tolerance (DBFT): it is an extension of the BFT approach where nodes are categorized as ordinary nodes and bookkeepers. Bookkeepers, elected by ordinary nodes, represent them in the consensus process. When a bookkeeper wishes to add a transaction, they broadcast it to the network, and at least sixty-six percent of other bookkeepers must validate it before appending it to the blockchain [52].

4.3.2 In permission-less blockchain

Commonly used protocols for public blockchain are [49]:

1. Proof of Work (PoW): in this kind of approaches, when a new transaction is submitted to the blockchain before being appended, nodes organize themselves into groups to solve a mathematical challenge. The group with a greater number of members will find the

solution more quickly, fostering collaboration among a larger number of participants and preventing the addition of erroneous transactions.

2. Proof of Stake (PoS): his approach simplifies consensus complexity by substituting the selection of miners based on proof of computational ability with a selection based on proof of ownership of some asset within the network. Nodes are rewarded for fulfilling tasks under specified conditions and penalized for misbehavior, ensuring that no node can join and continuously insert transactions.

4.4 Applications

Researchers have leveraged blockchain principles to enhance numerous classical security approaches. For instance, in the context of supply chain workflows, blockchain is utilized to ensure non-repudiation. Suppliers publish contracts describing the services or goods they offer, and interested parties adhere to these requirements, conducting transactions that are immutably recorded in the blockchain. This logic is applied at each node in the chain, from the initial supplier to the ultimate customer. Each intermediate element plays a dual role as both a producer and a consumer, resulting in comprehensive traceability of economic activities [51] [47].

Applications are exploring the improvement of zero-trust architectures through the use of blockchain. For instance, in [45], the authors introduced an approach based on reputation for block validation. The validation consensus threshold is adjusted based on the number of involved miners and their respective reputation. Blockchain has also been integrated into distributed collaborative intrusion detection systems. This direction aims to leverage blockchain for updating rules databases and to utilize consensus algorithms for detecting intrusions [53].

Blockchain was initially designed to ensure traceability and uphold privacy. The necessary information on participating nodes serves as a proof of integrity rather than identity. In this environment, nodes can operate while being identified based on their public keys, participate in transactions by adhering to the specified contracts, and verify the integrity of all operations by transparently accessing the information contained in the blockchain.

4.5 Conclusion

This section was dedicated to blockchain technology, it highlighted the elements crucial to the construction of our work, in particular by presenting a common classification of consensus protocols. However, it also addressed the layered model of a blockchain, its applications and typology.

5 Chapter conclusion

This chapter has introduced three key technological paradigms which were game-changers in the computer science research fields for the last decade. In broad outline, we have reviewed the definitions, components, architectural organizations, functional models and applications of each of these domains.

The Internet of Things provides a viable infrastructure as a global interconnection network on the one hand, and as a portal between the realm and cyberspace on the other. We benefit from this every time we need to automatically capture behavioral information about users or environment, or if we want to apply actions directly without human intervention.

As for the cloud, it provides considerable computing power to process the data collected by the IoT and provide useful decision-making information. Combining IoT and cloud computing produces what we call the cloud-centric Internet of Things, which will couple the ubiquity of IoT sensors and actuators on the one hand, and the high computing power of the cloud on the other, to create a unique synergy.

Blockchain is a prodigious way for data storing. Indeed, its organization as distributed ledger shared between participants, the collective validation of transactions and the integrated hash tree offer an unrivalled solution for guaranteeing the integrity and non-repudiation of the information it contains. In addition to being immutable and privacy-aware it integrates superbly well with IoT edge devices thanks to the lightweight protocols and data structures that can be embedded.

The next chapter will be dedicated to the state of the art. We will focus on levels were automation mechanisms play a central role. For example at the level of perception layer of IoT we will review leader election algorithms within wireless sensor networks (WSN) and at cloud computing environments we will study tasks scheduling and load balancing algorithms.

Chapter 2

State of the art of automation mechanisms and applications of Internet of Things and underlying technologies

1 Introduction

In the context of this thesis which has as main purpose automation mechanisms at both functional and application levels of the Internet of Things and the underlying domains, we have decided to focus on three problems inherent to the field. The way in which we present the state of the art is somewhat original, given that our work has a broad spectrum and is therefore not concentrated on a single fragment of the problem.

Leader election in the Internet of Things (IoT) is a vital mechanism for selecting a subset of nodes to manage others, a critical aspect for auto-organization given the vastness of the network where manual intervention becomes impractical. Given the nature of connected objects, this mechanism is designed to handle resource constraints. Numerous algorithms have been proposed to achieve leader election, taking into account factors such as node capabilities, data quality, energy efficiency, and network connectivity. A successful and robust leader election in the IoT context contributes to enhanced system reliability, scalability, and coordination, resulting in optimized resource utilization and improved overall performance. In the following, we will review some recent leader election algorithms and discuss their limitations.

Load balancing in cloud computing is a critical module responsible for optimizing resource utilization and achieving equitable distribution of workloads across servers. Its primary objective is to minimize the occurrence of under-loaded or over-loaded servers. In this overview, we will endeavor to highlight significant contributions categorized by their respective approaches and discuss their inherent limitations.

In recent years, interest in the problem of global warming has grown as an alarming sign of environmental and ecological danger. Researchers in computer science fields have come up with a number of platforms that enable people to assess their impact on the environment directly online and we're going to take a brief look at how these solutions work.

The rest of this chapter is organized in sections covering each a particular set of works proposed to deal with one of automation problem we worked on:

The section 2 is dedicated to solutions proposed to perform leader election in several fields of the Internet of Things. The most recent works on leader election in robotic networks, in virtual light traffic context and in wireless sensor networks are reviewed and briefly presented.

The section 3 summarizes all the works realized for ensuring load balance in cloud computing context within the last years and for each class of algorithm. Indeed the subsection 3.1 is dedicated to static methods, Indeed the subsection 3.2 summarizes dynamic algorithms and Indeed the subsection 3.3 presents hybrid approaches.

The section 4 describes briefly the existing digital ecological footprint calculators. Despite the preoccupations of governments and international organizations, the field is still uncharted and there is a real lack of proposals. This can also be attributed to the fact that the subject is still relatively recent.

2 Leader election

The principal challenges in any technology deployment and achievement are security services and automation mechanisms. Without establishing a dependable trust model between users and devices, achieving widespread integration is impossible [54] [55]. Numerous solutions have been proposed to ensure security services in a network context. For instance, when examining authentication protocols, it becomes evident that they are often based on a set of assumptions that imply the presence of a centrally reliable node, referred to as an authentication server, to authenticate a set of edge nodes [56].

It is important to note that these central nodes are manually selected from a group of computers or objects with high computational capabilities before deployment, and they remain unchanged during runtime without human intervention. However, this limitation needs to be addressed in modern networks that seek to enhance recovery speed from failures by minimizing the degree of human intervention. This is where mechanisms such as leader election become essential.

The leader election mechanism has played a central role in automation and self-management since the inception of distributed systems. It enables the designation of a global leader or a set of local leaders responsible for coordinating the work in a group of digital entities. This coordination can manifest in various forms, including task assignment to a set of nodes, distribution of network load, allocation of resources for job completion, and more [57].

Numerous studies have proposed leveraging the leader election mechanism to address various challenges in different application fields. Applications in distributed systems and ad-hoc networks, for instance, aim to select a node as a job manager to allocate tasks to other nodes [58] [59] [60]. In the context of the IoT, this mechanism finds application for diverse purposes [61] [62]. Some researchers tackle the Virtual traffic Lights (VTL) problem in the context of Vehicular Ad-hoc Network (VANet) by introducing an election phase to designate a car responsible for generating and broadcasting VTL [63] [64] [65]. Others apply this mechanism to choose a robot leader for managing exploration and military tasks without human intervention [66] or for selecting a protocol among population members [67].

The leader election mechanism is also applied in various use cases within the context of WSN. Firstly, it can be utilized to organize a set of sensors that are out of the range of any gateway. The mechanism selects one among them to coordinate and find a route to the nearest zone managed by a gateway. Ahcene B et al. proposed BROGO, an approach that starts by establishing a spanning tree from the initiator node to enable routing the value of each node to it. Subsequently, this root node takes charge of deciding which node becomes the leader by comparing received values and selecting the minimal one. It's important to note that the authors assume a flat network composed only of sensors [68] [57]. Secondly, in scenarios where random deployment leads to multiple gateways managing the same area, responsible for tasks such as authenticating sensors, the leader election mechanism can be employed. This approach aims to extend the lifespan of the WSN by electing a leader and putting others in a hibernation state until the elected leader fails.

However all these algorithms present the following drawbacks :

- The size of the network has a significant impact on the performance of the algorithm. All algorithms initiate the process by searching for a spanning tree from the root node, which serves as the initiator of the election process. The duration of this initial phase directly affects the overall execution time of the algorithm.
- The crucial role played by the root node introduces a single point of failure paradigm. In other words, if the root node fails, the entire protocol execution will fail, and the network will experience a delay before restarting it.
- These algorithms are grounded in the principle of maximal value, utilizing only one specific piece of information from each node, such as battery level or computational capacity. However, this approach overlooks the environment of sensors and gateways and is not logical if the algorithm is executed shortly after the deployment of a homogeneous WSN.

2.1 Leader election algorithms

Traditionally, leader election algorithms find application in distributed systems and collaborative networks to designate a coordinator responsible for autonomously distributing tasks and synchronizing results. The election process involves a combination of criteria, including computation and storage capacity, networking stability, physical position in the network topology, and more.

In the context of the IoT, leader election also serves numerous beneficial purposes. An example is the proposal made by Christoph Sommer et al. for self-organizing intersection management [63]. They attempted to introduce a leader election-based approach to achieve VTL, where vehicles approaching an intersection exchange messages to organize themselves and avoid collisions.

The objective of the Virtual Traffic Light Algorithm is to elect a leader for each intersection responsible for computing and disseminating a traffic light program to other cars. Three key assumptions make this achievable: (i) vehicles are equipped with networking devices such as IEEE 802.11p, (ii) each car is equipped with GPS (Global Positioning System) and supplemented by self-localization methods, and (iii) every car maintains a table of neighbors containing Identifier (ID) of nearby vehicles. The approach builds upon the work of Vasudevan et al. [58], which is an algorithm for dynamic ad-hoc networks that assumes: (i) a specific metric allows ordering nodes based on the distance to the intersection, (ii) each node has a unique ID for breaking ties, and the ID can be derived from the MAC Address, and (iii) every node keeps track of the identifier of the current election session.

The VTL algorithm works as follows:

- Upon entering the service area of an intersection, a car broadcasts an announcement message containing its distance to the intersection and its own ID. Initially, the initiator considers itself as the nearest car to the intersection and utilizes a timeout to wait for replies to the announcement.

- Using the shortest distance to the intersection, when the timeout expires, the initiator selects a leader and broadcasts a message to inform other cars.
- If a car, not currently involved in any election, receives an announcement message, it then joins the ongoing election session and responds with its own distance.
- If a car, already participating in an election session, receives an announcement, it joins the one with higher precedence determined by the election index or car ID in case of a tie.

This algorithm relies on a specific criterion, which is distance. This can be beneficial in a particular context, such as in an ad-hoc network of nodes competing to gain access to a critical zone, like road intersections. However, in other contexts, such as WSN, this approach becomes impractical due to the static and heterogeneous nature of nodes. Additionally, the assumption of fully reliable communications is not achievable in the WSN context.

The necessary architecture for this algorithm mandates that cars be equipped with wireless networking devices and localization mechanisms, such as IEEE 802.11p and Global Positioning System (GPS). However, it does not utilize crucial features when in motion, such as speed and acceleration. No mechanisms were proposed to ensure tolerance to message loss; when a message is lost, the Local Dynamic Map can be corrupted, potentially leading to accidents. On the other hand, the criterion for choosing the leader is the distance to the intersection, with a vehicle that is likely to leave it quickly triggering an immediate election restart.

The evaluation primarily focuses on the time taken for intersection crossing and does not address the duration of the election or the number of iterations.

Florian Hagenauer et al. [63] introduced an Advanced Leader Election for Virtual Traffic Lights. The algorithm follows these steps when VTL are active at an intersection:

- Vehicles broadcast data about their momentum.
- When a problem is detected as an imminent impact, a VTL is generated.
- The nearest car to the intersection is selected as leader and generates the VTL.
- Once done, a new election starts or the leader designates another one before leaving.

The authors assessed the performance of their algorithm using key criteria such as car density, travel time, and message loss rate. Once again, the criterion employed for the election is the distance to the junction, which may not be applicable in other contexts. The election times for different traffic density values are not provided.

In [64], the authors introduced a novel algorithm for the leader election process in the virtual traffic light protocol. Using Vehicle to Vehicle (V2V) communications, each vehicle broadcasts its position and speed. If a risk is detected, the concerned vehicles follow a protocol that begins by electing a lane-based leader based on proximity to the intersection. Subsequently, the leaders of lanes elect one of them as responsible for traffic lights. This leader is tasked with deciding

the order of vehicle movement. The leader can either halt or proceed when its lane displays a green light. If it hands over control before there are no vehicles waiting at red lights, the election process restarts. The authors compared and demonstrated that this VTL approach reduces junction crossing time compared to classical traffic lights for infrequent scenarios with 30 vehicles. However, no comparison with other VTL algorithms was presented.

Several solutions have been proposed to align with the WSN context. One example is BROGO (Branch Optima to Global Optimum). In this approach, Bounceur et al. [57] aim to create a lightweight leader election algorithm suitable for the self-organization of sensors.

BROGO works as follows:

- First, a Flooding Leaf Founding (FLF) algorithm is used to build a spanning tree, indicating its root and leaves.
- Then, each leaf routes sends a message to the root. This process is used to allow routing an optimal value for each branch to the root node which will determine the global optimum.
- In the final step, the root node send a message to the global minimum node informing it that it is the leader.

Upon closer inspection, two primary issues arise: (i) the consequences if the initiator node, the root of the spanning tree, fails, and (ii) the FLF algorithm relies on a straightforward message-acknowledgment mechanism; what occurs when a node is dormant or a message is lost.

BROGO envisions a flat WSN comprising solely sensor nodes. The process begins by establishing a spanning tree that facilitates communication between nodes. It presupposes that messages involved in the election procedure must traverse the root node, introducing a one-point-to-failure paradigm in this assumption. The revised BROGO [68] addresses this concern by incorporating a delay, employing the Wait-Before-Starting (WBS) procedure. If the root node fails to respond within the specified time, another node takes its place. In the event of a leader node failure, the election process must recommence.

In the "wait before start" procedure, each node identified by x must wait for a duration defined by $x * w$, where w should be sufficiently high to accommodate the prior process of informing all nodes. After this waiting period, if no message is received, the root is deemed to have failed, and a second node initiates the election process. While this revision provides a solution for root failure, employing this approach introduces significant delays and does not address the issue of lost or latent messages (the classical problem expressed as: how to distinguish between latency or loss of a message).

No details were provided in the discussion section regarding energy consumption, which was calculated based on the number of exchanged messages. Additionally, there is no further information on the duration of the election phases.

Bounceur et al. [61] introduced an algorithm based on a collection of local leaders. This method assumes an arbitrarily flat network. Upon deployment, a set identifies nodes with local minima values, designating them as roots (local leaders) to initiate the flooding process for building a spanning tree. When two trees intersect, the one with the superior value persists, and the other halts. After the algorithm runs for a sufficient duration, only one spanning tree remains, and its root is declared the leader. While this algorithm is straightforward, it does not consider specific constraints or real-world conditions.

Another instance of leader election in the IoT context is found in collaborative networks of robots. Pasquale Pace et al. [66] presented a Management and Coordination Framework for Aerial-Terrestrial Smart Drone Networks.

In numerous scenarios, collaborative efforts between aerial and terrestrial robots demand a leader to coordinate their tasks effectively. The missions, initially defined outside the group of drones, are distributed among the collaborative robots by the leader. Upon receiving a message from the headquarters, nodes broadcast their willingness to take on the coordinator role to others within their radio range. Subsequently, they collaboratively decide on the election of a leader. It's important to note that this leader is not absolute and undergoes changes based on specific criteria over time.

The proposition is based on three functions:

- Look-up: this mechanism is implemented in particular cases during the neighbor discovery phase, utilizing multi-cast communication to refresh data.
- The leader Election procedure: is employed for the initial leader election and is invoked again in case of leader failure. It relies on multi-cast communication, selecting the leader based on the criterion of maximum remaining charge, equivalent to the maximum ID
- Mission and task execution: utilized by the leader to delegate tasks to other nodes for the successful execution of a particular mission.

This algorithm also assumes a flat architecture where all devices have approximately the same capacities. It includes both aerial and terrestrial drones in the election process to appoint a coordinator responsible for distributing tasks for collaborative mission completion. The objective involves a high-mobility model with dynamic leadership features, presenting a significant challenge not encountered in the WSN context.The total time for the election procedure is approximately 1000 ms when the number of nodes reaches 10, which is relatively high for such a small number of participants and may not be scalable to contexts with thousands of nodes.

Mahendra K.M et al.[69] proposed a bio-inspired ant colony approach for leader election in the context of Cognitive Radio Network (CRN). In line with the utilization of ant colonies for leader election in WSNs, the authors extended this concept to CRN. They specifically address Secondary Users (SUs), with the leader responsible for monitoring communication channels used by Primary Users (PUs) and assigning available ones to SUs as needed. The performance measures of the algorithm lack conclusiveness, as the comparison was made against outdated approaches that are not suitable for similar contexts.

2.2 Conclusion

In this section we have introduced the main works existing in the literature which are actually dealing with leader election problem. Despite their application cases they all try to reach out a common objective by making a system as automated as possible. Generally they all start by building a spanning tree and end by choosing only one global leader, we were inspired by these approaches and revealed their limitations such that we can improve them and better match the objective.

The next section continues by building the state of the art of another part of automation problem in cloud-centric IoT. This time it will focus on load balancing in the cloud environment and highlights the essential elements in the major part of recent works within the three main categories of algorithms.

3 Load balancing

Our work focuses on the Load Balancing module, one of the critical mechanisms in the cloud environment, responsible for maintaining equitable workload distribution between servers and virtual machines (VM). This component plays a pivotal role in the operation of cloud services and adherence to Service-Level Agreements (SLA). Ensuring optimal utilization of hardware resources and a balanced workload distribution contributes to enhancing the overall system performance by reducing the makespan of essential jobs [70]. Numerous proposals have been put forward to meet the expectations of cloud service providers regarding load balancing. Upon reviewing the literature, it becomes apparent that these solutions can be categorized based on two criteria that influence the approach to workload distribution: (i) information on the environment, tasks, and resources, and (ii) the phase during which the balancing takes place.

Indeed, the first category, which we label as static, operates solely upon the reception of new tasks. It decides on task assignments based on a set of non-evolving information, such as task length and the physical capabilities of servers. In contrast, the second category encompasses dynamic approaches that operate continuously, considering information on the current workload on each server, individual makespans of virtual machines, and primitives like tasks or VMs migration. These dynamic approaches aim to maintain an optimal utilization rate of all servers over time. The last category encompasses approaches that combine static and/or dynamic methods, either with each other or with additional complementary mechanisms. The goal is to enhance the performance of load balancing and address common shortcomings [71] [72].

The load balancing problem can be approached in various ways, commonly formulated as a bin packaging problem, clustering problem, or even akin to a path-finding problem. Regardless of the formal modeling, we can describe the elements constituting the problem as follows: given a set of tasks and a set of resources organized into virtual machines and physical hosts, the challenge is to determine, for each task, which virtual machine it should be assigned to and which physical server should host that virtual machine. It is crucial to bear in mind that each server is resource-limited in terms of Central Processing Unit (CPU), Random Access Memory (RAM), storage, and Bandwidth (BW) [73].

Many approaches have been proposed in the literature for load balancing in the cloud environment. Subsections 3.1 to 3.3 will provide insights into some of the most significant ones. It is crucial, however, to first comprehend the categorization used. The classification of algorithms commonly encountered involves three main classes, as depicted below [71]:

- Static approaches [3.1]: this category relies on pre-existing information about jobs and the capabilities of servers/virtual machines to determine the task assignment policy.
- Dynamic approaches [3.2]: in contrast with static methods, dynamic algorithms integrate real-time information, such as workload on resources and utilization rates, to decide on task assignment strategies.
- Hybrid approaches [3.3]: hybrid approaches are obtained by mixing static and dynamic approaches to overcome the shortcomings of each. Furthermore, many researchers go further and hybridize load balancing techniques with fault tolerance or task scheduling mechanisms.

3.1 Static approaches

Static load balancing is a class of algorithms that allocates tasks to different resources without considering their current states. Depending on the applied policy, the algorithm will distribute new jobs equally or randomly over all treatment units, regardless of their actual workload [74].

Among the static load balancing methods, the min-min algorithm stands out. Its principle involves evaluating the execution time of each task beforehand and identifying the task with the shortest duration. Subsequently, the algorithm locates the resource with the minimum completion time to execute this task and assigns it. These steps are repeated until all jobs are completed. However, a major drawback of min-min algorithms occurs when the number of short tasks exceeds the number of long ones, leading to suboptimal resource allocation. Another category is the max-min approach, which addresses this drawback by scheduling larger tasks first. However, this method penalizes short tasks, resulting in increased waiting times for them [71].

Many enhancements have been proposed to improve the performance of static load balancing approaches and reduce makespan. For instance, in [75], Kokilavani et al. introduced an algorithm that initiates with a min-min phase, rapidly dispatching the shortest tasks to the most efficient resources. Subsequently, the algorithm evaluates the makespan of each resource, reallocates tasks from heavily loaded resources, and assigns them to resources with a shorter makespan. The LBMM approach, while simple in principle, demonstrated a reduction in overall execution time and an improved distribution of jobs.

In a different line of improvement, another group of researchers extended the min-min approach by introducing a novel algorithm that considers three crucial constraints in the cloud environment: quality of service, task priority, and cost of service. Their solution also starts with a min-min phase where short tasks are initially assigned higher priority. Subsequently, the algorithm reorganizes load balancing by incorporating these priorities along with the three

constraints, expressed as numerical values, to generate dynamic priorities for ordering all jobs across the available resources [76].

In the literature, there are approaches tailored for specific use-cases and based on meta-heuristics inspired by nature. A noteworthy example that encompasses both aspects is the work by Zhan et al. [77], where they employ a discrete Particle Swarm Optimization (PSO) for constructing a static load balancing algorithm to ensure the distribution of tasks in a cloud environment. They introduced adaptations to the functions responsible for updating personal and global bests, as well as velocity. These modifications enhance the performance of PSO in addressing this discrete problem, preventing it from getting stuck in local optima.

As powerful as these techniques may be, they have the disadvantage of being unable to adapt to the increasingly dynamic nature of cloud environments. Static approaches schedule tasks upon reception, relying on a logic independent of real-time workload distribution. This limitation prevents them from dynamically adjusting load distribution as the utilization of resources evolves. In essence, static load balancing performs well in cloud environments with reduced workload variability, which may not be suitable for scenarios with peak periods.

3.2 Dynamic approaches

Dynamic load balancing techniques comprise algorithms that incorporate real-time information about the utilization rate and remaining makespan on each server to determine the assignment of new jobs and the manner in which already scheduled tasks should be migrated. These approaches can be categorized into two main types based on their calculation mode: online or offline. In online mode, tasks are assigned as they arrive in the system, whereas offline mode operates in batches, with tasks being grouped and processed at predefined intervals [78].

Nature-inspired meta-heuristics emerge as an ideal solution for addressing the dynamic load balancing challenge. Through various stages of adaptation, researchers have elevated them to a dominant class of approaches for dynamic load balancing. It is crucial to note that before enhancing these algorithms, it is necessary to establish a suitable mapping between the algorithm's parameters and the cloud environment. Additionally, defining novel search functions is essential for the effectiveness of these algorithms in addressing the intricacies of load balancing in dynamic cloud environments [79].

For instance, the authors of [80] introduced an enhancement to the bee colony optimization algorithm to achieve dynamic load balancing. This approach integrates constraints to simultaneously avoid overloading and under-loading virtual machines, reduce makespan, and minimize the number of migration operations. The fundamental concept behind this enhancement involves using the standard deviation of processing time on each virtual machine (VM) as an input to the load balancing model. A threshold is defined to classify VMs into two groups: overloaded VMs modeled as honeybees and under-loaded VMs modeled as food. Given the dynamic nature of the approach, the deviation values are updated each time a new task is received.

On the other hand, Seyedeh et al. [81] have combined two meta-heuristic approaches to better align with cloud SLA requirements. Initially, they utilize a firefly-based algorithm to generate an initial population of potential task/resource assignments and then optimize it using an Imperialist Competitive Algorithm (ICA). Initially, two instances of the firefly algorithm are executed separately to find two assignments: one optimizing for the best makespan and the other for the best load balancing. The outputs of these heuristics are aggregated into a multi-objective function for the ICA algorithm, which incorporates both constraints and aims to produce a workload balance ensuring a minimized makespan.

Another approach, utilizing a bio-inspired meta-heuristic, is proposed in [82], where the authors frame load balancing as a clustering problem. This involves grouping sets of virtual machines on physical servers with specified CPU and memory capacities. Clusters are initially formed by randomly placing VMs according to a feasible distribution, and then they are updated based on their respective workloads to maximize remaining resources on each server while eliminating workload on weakly utilized ones. The authors applied the bat algorithm to optimize both global and local search for finding new cluster centers and speeding up convergence. This entails reviewing the set of VMs and physical servers that constitute each cluster to dynamically maintain equity.

Many other works have been proposed, attempting to create variants that improve specific performance criteria. This objective can be achieved by eliminating or adding constraints depending on the service-level agreement and the type of service provided. Additionally, it can be accomplished by modifying hyperparameters of known models or by proposing new fitness functions for meta-heuristics. For example, Dalia et al. [83] introduced constraints not commonly considered by other researchers in their algorithm system. They added complexity by addressing the simultaneous arrival of requests, prioritizing tasks, and assuming a deadline for each job based on the relative service-level agreement. A distinctive feature of this approach is that if the workload on a server does not meet the requirement for correct execution of a given task, it is migrated to another server. Therefore, to achieve efficient workload distribution, the authors integrated load balancing and task scheduling within the same algorithm. Another example is provided in [84], where the authors present a variant of a genetic algorithm that considers the performance degradation of VMs during migration time and its impact on task execution.

In [85], the authors delved deeper into the integration of key constraints by incorporating elasticity into their model. Their architecture supports hardware proactive horizontal scale-up. After assigning tasks, a component of the resource broker monitors activity on servers and estimates whether there are scheduled tasks that will exceed their deadline. It then decides to create new virtual machines to balance the workload.

It is noteworthy that while these meta-heuristics form the backbone around which most load balancing algorithms in the cloud are built, other methods exist. These alternative approaches tackle the problem through different modeling or statements and leverage the power of various mathematical techniques.

In the literature, fuzzy-based approaches are frequently encountered for tasks scheduling and load balancing. For instance, in [86], the authors introduced a fuzzy-based algorithm for multidimensional resource planning, with a specific emphasis on file-sharing services in a cloud environment. The approach involves three stages: (i) collecting requests from users, (ii) utilizing trapezoidal fuzzification and fuzzy square inference for multidimensional resource scheduling, and (iii) designing a queuing network for the assigned tasks and resources.

Methods based on machine and deep learning are also prevalent, such as the one introduced by Zhao et al. [87], where they combine a Q-learning approach with a neural network. Initially, the scheduling plan is represented by a directed acyclic graph, with nodes characterized by a quaternion comprising a specific task, the execution cost, communication cost, and edges denoting relationships with successor tasks. The dynamic scheduler, before planning a workflow by distributing its jobs on virtual machines, invokes the algorithm to evaluate the execution scenario modeled as a graph. It applies a reward function aiding decision-making by emitting an action for the scheduler to implement. Another hybridization with a meta-heuristic method is proposed by Jena et al. [88], where a particle swarm algorithm is combined with a Q-learning approach. This Q-learning approach is used to adjust the velocity of particles and global bests to achieve quicker convergence towards an optimal load balancing solution.

Dynamic approaches are designed to adapt to real-time changes in constraints, ensuring an ongoing and fair distribution of workload while actively balancing the load flow based on the resource usage of each host. However, these approaches may introduce latency at the start of the load balancing process and are constrained by reaction times, even when dealing with a small number of tasks.

3.3 Hybrid approaches

Hybrid load balancing integrates both static and dynamic approaches, harnessing the advantages of each category to address the limitations of the other. While static methods provide a swift initial distribution of tasks, dynamic algorithms ensure ongoing optimal workload balancing. Hybridization extends beyond this combination and can involve integration with additional mechanisms, such as fault tolerance or dynamic task scheduling.

Bio-inspired meta-heuristics play a pivotal role in hybrid approaches. The literature on hybrid load balancing is abundant with examples, such as the proposal by Marwa et al.[89]. In this work, the authors combined the swarm intelligence of bee and ant colonies to create an osmotic hybrid optimization load balancing algorithm. Following an initial random distribution of jobs, an artificial bee colony is employed to swiftly identify overloaded and under-loaded servers. Subsequently, an ant colony is utilized to determine the optimal migration scheme for virtual machines among osmotic servers.

In another study [90], a team of researchers chose to blend ant colonies with fuzzy models. They incorporated a fuzzy module alongside Ant Colony Optimization (ACO) to assess the quality of the migration pattern obtained. In a nutshell, the fuzzy component is employed to update the pheromone traces, expediting the convergence process towards an optimal solution.

In [91], the authors advocated the combination of a genetic algorithm and the gravitational search method to amplify the searching process and diminish computational costs. The enhancement is achieved through a hybrid method for calculating particle positions at each step, employing a crossover technique in tandem with a gravitational constant function. Similarly, in [92], the authors fused a queuing model for virtual machine management with a crow search-based approach to optimize task placement, concurrently minimizing time wastage and energy consumption.

In [93], researchers extended their load balancing approach by integrating various reactive and proactive fault tolerance techniques with an expedited decision-making process for swift recovery. At the core of their method lies a dynamic scheduling approach, emphasizing replication as a fundamental constraint. Similarly, Haoran et al. [94] proposed a comparable integrated approach by incorporating fault tolerance and task scheduling into their model to enhance load balancing. The authors concentrated on hybrid real-time tasks categorized as data-intensive, process-intensive, or balanced tasks. By applying the same categorization to virtual machines, they increased the likelihood of improving system performance and facilitating the scheduler's operation. Moreover, they took a step further by combining checkpoint and primary backup techniques to formulate the recovery policy and corresponding task description. Ultimately, the resulting task list, including redundancy, was added as constraints to the scheduler.

In [95], Huaiying et al. put forward an approach aimed at ensuring quality of service in edge-cloud environments by integrating fault tolerance with task planning to maintain balanced load distribution. The authors improve upon the conventional primary/backup fault tolerance method by introducing QoS constraints, including time-based constraints. The primary and copy tasks are scheduled using a dedicated method that incorporates an adjustment procedure ensuring the placement of copies in a way that reduces both recovery time in case of failure and overlapping during normal operation. Another example of this hybridization approach is found in [96]. In addition to task planning, the authors propose a solution for monitoring activities on virtual machines forming logical clusters over physical hosts. They utilize metrics based on previous server performance to enable the system to proactively anticipate deviations and behaviors that do not adhere to specifications, facilitating a quick recovery from the last consistent checkpoint.

In [32], the authors employ machine learning techniques to optimize resource utilization, addressing both horizontal and vertical load balancing. They train an agent using a custom reinforcement learning approach, rewarding the agent based on the desirability of selected actions such as task assignment on a specific virtual machine or migration to another host. It's important to note that our list is not exhaustive, and numerous other works in hybrid load balancing exist. For instance, [97] focuses on dynamic resource provisioning tailored for a specific application in the intensive processing of meteorological data flows.

As demonstrated, hybrid approaches effectively address the limitations of isolated static or dynamic techniques. They offer advantages such as increased responsiveness, better constraint

management, and faster task distribution. However, it's important to acknowledge that the implementation of hybrid approaches can be complex.

3.4 Comparative analysis

The table 2.1 introduces a comparison of the most recent and relevant approaches from the state of the art presented in this section. The comparison is based on structural criteria, including:

- The nature highlighting the type of the approach (static, dynamic, or hybrid).
- Combination with any other mechanism such as fault tolerance or task scheduling.
- The fundamental method or metaheuristic on which the approach relies.
- The advantages the approach provides.
- The limitations it suffers from or how it is presented suffers.
- The performance metrics employed to validate and evaluate the approach.

3.5 Conclusion

In this section we have given a taxonomy of load balancing algorithms. We have introduced the static ones characteristics, major contributions and limitations, then we moved to dynamic algorithms which are more reactive and take into account real-time information on servers and cloudlets but are slowly since they mainly rely on meta-heuristics. We have ended it with hybrid methods which take advantage of combination between static and dynamic algorithms, or some times literally combination between load balancing and other mechanisms to come over limitations of the two first families.

The next section will introduce digital ecological footprint calculators which are platforms allowing people and companies to declare their activity and to estimate approximately their carbon footprint .

Approach	Nature	Combination	Core method	Merits	Limitations	Metrics
[75]	Static	No	Min-Min	Fast	Not adaptative	Makespan
[77]	Static	No	Particle Swarm Optimization	Fast	Not adaptative	Makespan
[80]	Dynamic	No	Bee colony	- Fast - Semi-adaptative	VM level migrations	- Makespan - Migrations
[83]	Hybrid	Task scheduling	- Opportunistic load balancing - Min-min	- IaaS level - Optimize resource	Ignore common constraints	- Makespan - Resource utilization
[82]	Hybrid	Task scheduling	Bat fly algorithm	Considers pricing	Ignore common constraints	- Makespan - Flow time - Resource utilization
[98]	Dynamic	No	Genetic algorithm	Fast	Lack of performance evaluation	Performance degradation
[85]	Hybrid	Task scheduling	Custom method	- Fast - Task-level migration - High scalability	Not adaptative	- Makespan - Migrations
[81]	Hybrid	-Task scheduling - Sort of fault tolerance	- Imperialist competitive algorithm - Firefly algorithm	- Fast - Considers linked tasks	- Complexity - Lack of performance evaluation	- Resource utilization - Makespan
[86]	Hybrid	Task scheduling	Fuzzy-based logic	- Very-responsive - Fast - Resource optimization	- Complexity - Lack of performance evaluation	Resource utilization
[87]	Hybrid	Task scheduling	Deep Q-learning	- Highly-adaptative - Scalability	- Complexity - Lack of performance evaluation	Makespan
[90]	Hybrid	Task scheduling	- Ant colony algorithm - Fuzzy-based logic	Highly responsive	- Complexity - Lack of performance evaluation	Response time
[89]	Dynamic	No	-Ant colony algorithm -Bee colony algorithm	- Enhance QoS of service - Optimize energy consumption	- Complexity - Lack of performance evaluation	- Energy consumption - Migrations
[93]	Hyrbid	Fault tolerance	Custom method	Optimize cloud resource utilization for real-time application	- Complexity - Lack of performance evaluation	Makespan
[96]	Hybrid	Fault tolerance	Byzantine fault tolerance	Synchronous checkpointing which keeps a global job consitency	- Ignore common constraints - Slow	- Makespan - Energy consumption - SLA violations

Table 2.1: Comparison of load balancing algorithms

4 Digital ecological footprint calculators

The 2022 report from the Intergovernmental Panel on Climate Change highlighted that human activities resulted in a net emission of fifty-nine gigatonnes of greenhouse gases globally in 2019 [99]. Despite numerous commitments by nations, attempts to stabilize the impact of global warming, driven by increased industrialization and hydrocarbon use, have proven unsuccessful [100]. According to a World Health Organization report, the impending drought is expected to lead to the migration of millions of people in the coming years [101]. It is evident that climate change will pose the most significant challenge for humanity in the decades ahead. To mitigate carbon dioxide CO_2 emissions from industries and individual activities, states play a crucial role in enacting regulations and implementing strategic planning [102].

Field	Category	Features
Individual	Transportation	- car trips - public transport - airplane travels
	Energy and water	- home consumption - hobbies - heating and air conditioning
	Food	- organic and farmed foodstuffs - meats - ration of locally produced food
	Health & education	an overall average expressed as the total amount of carbon dioxide generated by utilities relative to the number of citizens
States & campanies	Infrastructure & buildings	- lifecycle of buildings and infrastructure projects - transportation infrastructure maintenance - building and evolving energy production structures
	Supply chain	- energy and food supply - importing raw materials and transport of finished products
	water pollution and consumption	- estimation of the volume of water consumed in production or construction processes - polluted water rate by industrial waste
	deforestation	- cost in green space of each project - impact on living beings (human or not) and neighbouring ecosystems

Table 2.2: Commonly used features for ecological footprint calculation

4.1 Digital ecological footprint calculators

Various digital ecological footprint calculators have been introduced to assess the environmental impact of individuals and companies [103]. However, a common limitation is that they rely on reported information rather than directly collecting behavioral data on activities. Enhancing regulatory mechanisms by directly tracking the behavior of companies and individuals can provide more accurate insights for informed decision-making by states and help individuals understand their true environmental impact. Nevertheless, a significant challenge arises in balancing activity tracking with privacy and confidentiality considerations.

To the best of our knowledge, this is the first instance of such a method being proposed. Existing literature lacks approaches that enable the calculation of an ecological footprint based on directly collected data during the activities of both companies and individuals. Over the last decade, numerous research papers have addressed the ecological emergency, presenting various methods for calculating carbon footprints and emphasizing the impact of using such calculators on societal behavior. Table 2.2 provides a summary of the commonly requested information by these calculators, typically presented in the form of web platforms.

Mulrow et al. [103] conducted an assessment of the design and user interaction patterns of digital footprint calculators, emphasizing those dedicated to individuals. They introduced a feature index, concentrating on two key concepts: (i) depth of inputs, which measures the variety of categories of collected information and the level of detail for each category; (ii)

user engagement, assessing display quality and the intuitiveness of user interaction. In their review of thirty-one calculators, none were found to be behavioral, reinforcing the absence of calculators similar to ANPA-GIEFC. Another study by Jiayi, Qiuchen, & others [104] focused on a more specialized comparison of calculators dedicated to buildings and infrastructure. This focus was driven by the fact that one-third of global energy consumption is attributed to these sectors. Their specialization allowed for a more in-depth exploration of features used, especially in evaluating the environmental impact of different life-cycle stages in building processes.

After extensive searches on reference engines such as Google Scholar, it was found that digital ecological footprint calculators are scarce, primarily adopting a declarative approach and often focusing on individual footprints.

4.2 Conclusion

This section was dedicated to digital ecological footprint calculators. As it is easy to see, there is a real lack of works in this field that attempt to estimate the ecological footprint of individuals and companies. It is also important to note that they are all declarative and therefore not based on real activities or behavioral data which makes them very imprecise.

5 Chapter conclusion

This chapter was dedicated to the state of the art, it has given the opportunity to introduce major works related to ours. We namely introduced some important leader election algorithms, presented three families of load balancing algorithms and finally summarized existing digital ecological footprint calculators.

The leader election problem represent a key point in all automated or self-organized systems, we have shown that this is an active research field where lot of contributors have given lot of improvement in many application domains like virtual traffic lights, robotic networks or moreover in perception layer of the Internet of Things.

The load balancing and tasks scheduling are the Swiss-knives of the cloud automated environments, their performance directly impact quality of service (QoS) and thus act on the service-level agreement (SLA). Lot of approach exists which are stateless like static ones which encourages speed on QoS, others are stateful and take in consideration real-time information on their environment. A last category stand for a hybridization of the two previous one and tries to come over their drawbacks.

Calculating one's impact on the environment directly is an impossible task, so digital tools have been created to help individuals and companies calculate it and track their activities in an attempt to reduce their impact.

The next chapter will focus on our own contributions. It will show how we proceed to reduce leader election complexity in flatten networks environment like in wireless sensor network context. It will then detail our multi-level hybrid load balancing and tasks scheduling algorithm which rely on meta-heuristics and clustering to perform better workload distribution in cloud environments. It will end by introducing the global privacy aware infrastructure for an automatic global ecological footprint calculation.

Chapter 3

Contributions

1 Introduction

The preceding chapters have enabled us to respectively introduce the theoretical elements needed to understand our thesis work, and then to position ourselves among existing solutions known in the literature. This chapter will be devoted exclusively to the presentation of our research work carried out during this thesis.

We have proposed a fully distributed, fault-tolerant leader election algorithm, which outperforms its competitors. This is due to a reduced complexity since it removes building spanning tree step which is a common initial phase in election process. Moreover it avoids the need to regularly re-run the election algorithm, as it prepares a list of substitutes who are designated to take over in the event of the leader's failure [6].

We then proposed a new hybrid mechanism that enables task scheduling and load balancing in a cloud environment that operates on two levels. Indeed, it improves workload distribution among servers and reduces makespan through a multi-step procedure: k-means-based clustering, job scheduling on clusters using round-robin, then job scheduling on servers within the same cluster using a genetic algorithm, and finally a multi-level load balancing function. Although it has a complex structuring, this mechanism enables a high degree of decoupling in the functions of its modules that make and which cooperate to complete its missions Thus reducing overall task execution times and better meeting the constraints of service-level agreement with clients [7].

We end by presenting our digital footprint calculator. It is as best as we know the first contribution of kind since it relies on behavioural and not declarative data to calculate the impact score. It is feasible because it is built upon existing technologies which are the Internet of Things and the blockchain and it preserves privacy. A public key infrastructure is used to improve the infrastructure security and allow all actors (customers or producers) to interact in a reliable and anonymous trackable manner [8].

We have undertaken a comparative analysis of machine and deep learning techniques for detecting botnet-based attacks in the Internet of Things (IoT) [105]. As this study aligns more with a security-focused approach, we deemed it as beyond the scope of this thesis and have chosen not to incorporate it in this work.

This chapter is organized as follows: first the section 2 presents the distributed and reliable leader election framework for wireless sensor network. Then the section 3 introduces in details the novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment. It end by presenting A novel privacy-aware global infrastructure for ecological footprint calculator (ANPA-GIEFC) based on Internet of Things and blockchain in the section 4.

2 Distributed and reliable leader election framework (DRLEF)

In this section, we present a leader election algorithm designed to choose a subset of gateways to serve as local leaders responsible for authenticating and managing sensors within their range. To overcome the challenges discussed in Section 2, we introduce a novel algorithm named Distributed and Reliable Leader Election Framework (DRLEF). This innovative approach aims to address the limitations found in existing solutions and offers the following advantages:

- Utilizing multiple gateways within each area to mitigate the single point of failure scenario.
- The centrality criterion significantly influences the leader designation process.
- An algorithm is employed to compute disjointed lists, identifying broadcast areas without intersections. The union of these areas equals the set of all sensors within the reach of the competing gateways, enabling the transmission of election messages to each sensor only once.
- The election result provides a list of candidates to assume leadership in the event of a failure by the elected leader.
- DRLEF has the potential for generalization to various use cases due to its applicability in hierarchical systems and reliance on standard criteria. Minor adaptations of parameters and thresholds may be adequate to make it suitable for other scenarios.

This section is structured as follows: subsection 2.1 presents assumptions and used notations. Then 2.2 is devoted to our algorithm. Subsection 2.3 is dedicated to simulation and implementation method. Finally 2.4 discusses the obtained results.

2.1 Assumptions and notations

We begin by enumerating the assumptions that render our algorithm feasible, followed by a step-by-step presentation of the main phases of DRLEF. Table 3.1 outlines the variables employed by DRLEF along with brief descriptions.

For our implementation, we assume a classic WSN composed of sensors with limited calculation capacities and gateways which are in charge of data synchronisation, routing and local management. We assume the following:

- WSN are ramdomyl deployed.
- Nodes are not highly mobile.
- A constant ratio $Sensors\_number/Gateways\_number$ is maintained.

Variable	role
gws_i	Gateway indexed i
List of Direct Neighbors: Nodes (LDNN)i	Set of sensor nodes in radio range of Gateway (GW)i
Number of Direct Neighbors: Nodes (NDNN)i	Lenght of LDNNi
List of Direct Neighbors: Gateways (LDNG)i	Set of Gateways in radio range of GWi
Number of Direct Neighbors: Gateways (NDNG)i	Lenght of LDNGi
Election Initialization Message (EIM)	Election Initialization Message
DEVij	Deviation of a \GWi to a GWj in its radio range
Election Concerned Gateways (ECG)	Election Concerned Gateways List
Start Election Message (SEM)	A message sent by the Central Gateway to inform participants that they can start the election process
Candidacy Message (CM)	A message sent by a Gateway to the central Gateway to notify that it participates in the election
Active Status Message (ASM)	A periodic message sent by the elected gateway to its relay to indicate that it remains active.

Table 3.1: DRLEF Variables and messages

An essential concept that we leverage in the DRLEF algorithm is centrality. Node centrality is a measure of the importance of a node in the network, particularly crucial in the context of the Internet of Things (IoT) where edge nodes play pivotal roles. The most commonly used methods for researching node centrality can be classified into the following categories:

1. Between-ness: nodes are assigned a score calculated as the fraction of the number of shortest paths passing through a specified node relative to the total number of shortest paths. The higher the ratio, the more central the node is considered.

2. Closeness: A node is deemed to have higher closeness if the sum of its shortest path distance to all other nodes is smaller.

3. Degree: takes into account the number of direct neighbors of a particular node.

4. Local Fielder Vector Centrality (LFVC) [106]: measures the network's sensitivity to specific node deletion.

5. Others: a lot of other approaches exist like Eigenvector centrality or more over the Ego Centrality method.

2.2 Algorithm and main steps

The DRLEF algorithm for leader election aims to designate a set of Gateways to coordinate the WSN, with other gateways in the same radio range serving as hibernating candidates. These candidates will wake up in a specific order to replace the designated leader in case of failure. The complete algorithm and its main phases are outlined in Algorithm 1.

2.2.1 Exploration phase

This follows a classical approach to construct a local network map. Each gateway initiates an exploration message, and the gateways or sensors within radio range respond with a message containing their network parameters. Subsequently, each gateway creates two lists of direct

Algorithm 1: DRLEF Algorithm

Data: Randomly deployed WSN
Result: Leader per area and ordered successor lists
`// Exploration phase`
1 **foreach** $gws_i \in list_{gws}$ **do**
2 | apply a classical flood procedure to detect the direct neighbors of gws_i from its area;
3 | **if** *gws_i is alone and there is no another one around* **then**
4 | | leave algorithm;
5 | **else**
6 | | go to initialization phase (*line* 9);
7 | **end**
8 **end**
`// Initialization phase`
9 **foreach** $gws_i \in list_{gws}$ **do**
10 | gws_i creates an EIM (election initialization message) containing its identifier and a list of its neighbours gateways and sensors;
11 **end**
12 The gateway with the maximal number of neighbouring gateways is considered as CGW (Central) and is in charge of computing the deviation of each neighbour gateway; `// the deviation is the average of deviations in the number of neighbours sensors between the gateway and the other gateways in the same area`
13 The CGW adds gateways with a deviation greater than a defined threshold to a list called ECG (Election Concerned Gateways);
14 The CGW sends a SEM (Start Election Message) containing ECG;
15 **if** *the number of gateways in ECG and in radio range of each other is greater or equal to two* **then**
16 | go to election phase (*line* 20*);*
17 **else**
18 | leave algorithm;
19 **end**
`// Election phase`
20 **foreach** $gws_i \in ECG$ **do**
21 | **if** *gws_i has the minimal score* **then**
22 | | the gateways in the same ECG reduces the score of gws_i by the score of the other gateways;
23 | **end**
24 | gws_i compares its score to those of other gateways in the same ECG;
25 | gws_i sends to the CG (Central Gateway) a CM (Candidacy Message) containing its final score;
26 **end**
27 Based on the received scores, the CG computes an ordered list of candidates of which the first element is the local leader;
`// Failure tolerance phase`
28 **repeat**
29 | the elected gws sends an ASM (Active Status Message) to the next gateway gws_{next} in the ordered list which will stay in hibernating mode;
30 **until** *there is an interruption for a certain delay*;
31 gws_{next} sends an EM (Elected Message) to the other gateways in the same list to inform them that it takes over;

neighbors—one for sensor nodes and the other for gateways. It is important to note that if a gateway is the sole one in its radio range, it is automatically considered the leader in that network segment.

This phase is summarized by Algorithm 2.

Algorithm 2: Network Exploration

Data: Randomly deployed WSN
Result: Gateways with neighbor lists
foreach $i \in N$ **do**
  //N set of gateway indices
  GW_i sends exploration message to direct neighbors
end
foreach $i \in N$ **do**
  GW_i builds $LDNN_i$, $LDNG_i$; `// List of Direct Neighbor: //sensor nodes (LDNN) and Gateways (LDNG)`
end
if GW_0 *has no gateway as neighbor* **then**
  GW_0 is a local Leader;
  Stop the Algorithm;
end

2.2.2 Initialization phase

‘

This phase, along with the subsequent ones, takes place when two or more gateways are within radio range of each other. In the initialization phase, each of these gateways prepares the necessary lists and information for the election phase. The steps of this phase are as follows:

- Each gateway prepares an `Election_Initialization_Message` containing its own identifier along with the two lists previously built. Subsequently, it sends this message to all gateways within its radio range.
- Upon receiving all messages, each gateway checks if it has the maximum number of gateways as direct neighbors. If not, it enters a hibernation state and awaits further messages.
- Otherwise, the gateway that has the maximum number of gateways as direct neighbors is identified as `Central_Gateway` (CGW), It is crucial to distinguish this central gateway, chosen for intermediary calculations, from the elected one, which will be designated to lead its zone of the WSN at the algorithm's conclusion.
- The CGW calculates deviations for each gateway in comparison to others. The local deviation for a gateway is then computed as the average of all individual deviations of that node. The goal is to identify gateways with a high average number of sensors differing from others. If this value exceeds a threshold, indicating that if the gateway hibernates,

a significant number of sensor nodes will be unreachable, an `Election_Abort_Message` is sent to these gateways, rendering them ineligible to participate in the election. On the contrary, if the deviation is within a reasonable range, the concerned gateway is added to the `Election_Concerned_Gateways` (ECG) list, ordered according to $NDNN_i$, allowing them to be candidates for leader election.

- The CGW broadcasts a `Start Election Message` (SEM) containing a keyword to initiate the election process and the ECG, which will be utilized by each GW_i to compute its final list of direct neighbors (sensors), to all gateways in the ECG.

Algorithm 3 Describes the operations of this phase.

Algorithm 3: initialization phase

Data: Gateways with local knowledge
Result: Local lists of election participants

foreach $i \in LDNG_i$ **do**
  EIM_i={GID, $LDNN_i$, $LDNG_i$}; `// GID: Gatway-Identifier`
  foreach $j \in LDNG_i$ **do**
    Send(EIM_i,j)
  end
end
`// After a waiting delay`
if $NDNGi \neq MAX(NDNG_j)$ **then**
  Wait for a message reception;
else
  `// The concerned Gateway is noted CGW as local central one`
  DEV_i=$AVGDEV_{ij}$; `// The local deviation(`DEV_i`) of each Gateway is then given as the Average of` DEV_{ij}
  if $DEV_i > Threshold$ **then**
    Send(EAM,i); `//` EAM`: Election Abort Messages`
    `//` GW_i `must stay on`
  end
  $ECG = ECG \bigcup \{i\}$;
  `//` ECG`: Election Concerned Gateways`
  send(SEM,ECG); `//` SEM`: Start Election Message`
  }
end
if *there are 2 or more gateways in radio-range of each other* **then**
  Goto phase 3 ;
end

2.2.3 Election phase

Once all lists and candidates are prepared, the election phase can commence, at the conclusion of which the leaders will be determined. It is outlined in Algorithm 4, and the steps can be described as follows:

- On each gateway receiving SEM:

- Initialize the Final List of Direct Neighbors (LDNF_i, sensor nodes) to the initial one (LDNN_i).
- For each GW_j from ECG, the GW_i checks if the $\text{NDNN}_j > \text{NDNF}_i$ then it concedes the common sensor nodes to GW_j and $\text{LDNF}_i = \text{LDNF}_i$ - LDNN_i.
- At the end we assume $\text{NDNF}_i = |\text{LDNF}_i|$, each gateway checks if its NDNF_i is greater than Threshold, it sends a candidacy message to the CGW containing its Identifier and NDNF_i.
- When receiving the first CM_i, the CGW waits a predefined laps time for receiving other CM_i and does not accept anymore Candidacies after this delay.
- It creates then an ordered list `Elected List`(EL) according to NDNF_i and attributes rank to each GW_i.
- The GW_i which has the first rank is the elected one (noted EGW) and will stay on.
- CGW sends the EL to all concerned GW_i.
- Other GWi sends a sleep command to sensor nodes in their LDNFi and activate hibernate mode.

Algorithm 4: Election phase

Data: Local Lists of Election Participants
Result: Local Leaders

for $i \in \{1, \cdots, N\}$ **do**
  $LDNF_i = LDNN_i$;
  for $j \in \{1, \cdots, Len(ECG)\}$ **do**
    if $NDNF_j > NDNF_i$ **then**
      $LDNF_i = LDNF_i$ - $LDNN_j$;
    end
  end
end
$NDNF_i = Len(LDNF_i)$;
if $NDNF_i > Threshold2$ **then**
  $CM = \{GID, NDNF_i\}$; // CM: Candidacy Message
  Send(CM, CGW); // CGW: Central GateWay
end
// On CGW when all CM received Estimated by waiting delay from first message receiving event
// Create an ordered list according to $NDNF_i$ in descending order
$ElectedList = \{(GID_1, Rank = 1), \cdots, (GID_n, Rank = n)\}$;
Send($ElectedList$, GW_i);
if $GW_i.rank = 1$ **then**
  $ElectedMessage = \{GID, Statut =$'Elected'$\}$;
  Send($ElectedMessage$, $LDNF_i$);
end
$HibernateMessage = \{GID, Statut =$'Not_Elected'$\}$;
Send($HibernateMessage$, $LDNF_i$);

2.2.4 Failure tolerance

This phase introduces a crucial mechanism that prevents the need for a complete restart of the algorithm when the leader fails. It can be broken down into the following steps:

- The CGW periodically transmits an `Active_Statut_Message` (ASM) to the next GW_i in the Elected List, to inform it that it is always active.
- If the second gateway does not receive a message within a predefined interval of time from the last ASM, it wakes up and sends K periodic `Status_Check_Messages`. If the CGW does not respond, the second gateway then sends an elected message to gateways and sensor nodes within its radio range.
- Then the last gateway takes over and become the leader, the fault-tolerance function is then restarted with the following gateway in the same list.

Algorithm 5: Failure tolerance phase

Data: A local leader
Result: a recovery on failure without re-election
// Current Local Leader is noted EGW as elected gateway
// ASM is an Active Status Message used to inform VGW that EGW is still active
// Vice gateway, VGW is one with rank next to EGW
// On EGW:
1 **while** *True* **do**
2 | $EGW.wait(T)$; // T is a prefixed delay
3 | Send(ASM, VGW);
4 **end**
// On VGW:
5 **while** *Receiving ASM within* T **do**
6 | Stay in hibernate mode ;
7 **end**
8 $VGW.wakeup()$;
9 $i = 0$;
10 **while** *(*$i < k$*) and (no ASM received)* **do**
11 | Send(CSM,EGW) ;
12 | $i = i + 1$;
13 **end**
14 **if** *receiving ASM* **then**
15 | Restart Failure tolerance phase ;
16 **end**
17 $EGW = VGW$;
18 $ElectedMessage = \{GID, Statut = "Elected"\}$;
19 Send($ElectedMessage$, $LDNF_i$);

2.3 Implementation

We will present the environment of our simulation, the main results of experiments and a brief comparison with existing solutions.

Parameter	value
Nodes	200 - 1000
Radio Range	100 m
Ratio —Sensors—/—gateways—	0.1
Routing protocol	Routing Protocol for Low-Power and Lossy Networks (RPL)
Threshold	0.2
Threshold2	10

Table 3.2: DRLEF Simulation Parameters

We have implemtend DRLEF in the JBOTSIM framework with the given parameters in Table 3.2 on a machine with the following characteristics:

- Processor: Intel®Core ™i7-4600U CPU @ 2.10GHz 2.70GHz.
- RAM: 8 GO.
- OS: Windows 64 bit, x64-based processor.

JbotSIM is a Java library that facilitates the description, execution, and evaluation of distributed algorithms. Additionally, it provides a graphical interface for visualizing simulation scenarios in real-time [107].

Various criteria have been employed in the literature to assess the effectiveness of election algorithms, with some being tailored to specific contexts. For instance, Pasquale Pace et al. [66] assessed their management and coordination framework for aerial-terrestrial smart drone networks based on the average election session duration. We consider this criterion to be optimal for standardizing simulations and enabling the comparison of different approaches, irrespective of specific scenarios.

We implemented our algorithm and conducted multiple scenarios with varying numbers of sensors (nodes), while maintaining a constant number of sensors per gateway at 10. To obtain more realistic estimates of election duration, we repeated the experiment 50 times for each value.

2.4 Results discussion

The results of simulation are summarized in Table 3.3 which gives duration by step and total time for each configuration.

	200	400	600	800	1000
Exploration	6.36	17.19	39.7	104.3	233.1
Initialization	0.7	1.4	5.4	3.1	15.3
Election	11.7	157.3	573.2	1965.1	5032.4
Total	18.76	263.86	618.3	2072.5	5280.8

Table 3.3: DRLEF average election session duration

Each row corresponds to a distinct stage of our algorithm. The initial cell in each column indicates the number of sensors for the respective simulation scenario. Throughout our experiments, we maintained a consistent ratio of 1 gateway for every 10 sensors.

Figure 3.1 to 3.4 present time evolution according to the number of nodes during the algorithm phases.

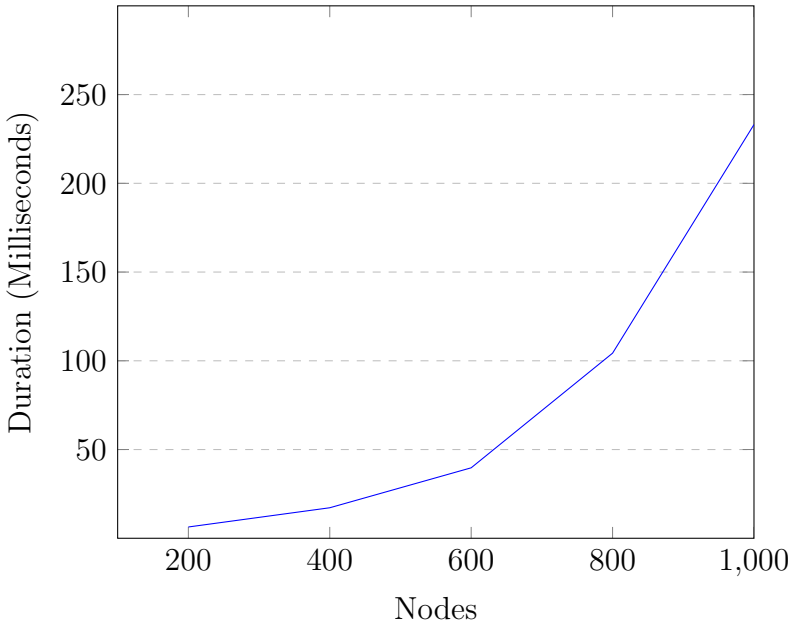

Figure 3.1: DRLEF exploration phase duration

Results for exploration phase from Table 3.3 are shown in Figure 3.1. The curve adheres to the shape of a second-degree polynomial, and this observation holds true for the other phases as well. Being entirely distributed, our approach maintains scalability without incurring a notable increase in computations and, consequently, time. As a refresher, the initial phase is exploration, which involves the exchange of messages. During this phase, gateways dispatch requests and await responses to construct a direct neighbors list comprising other gateways and sensors.

Figure 3.3 shows the Election phase results described in Table 3.3, this phase consumes the majority of the total time. All calculations to designate the set of gateways elected as local leaders are performed during this step. This is based on direct neighbors found in phase 1 and using lists prepared and exchanged via messages during the second one.

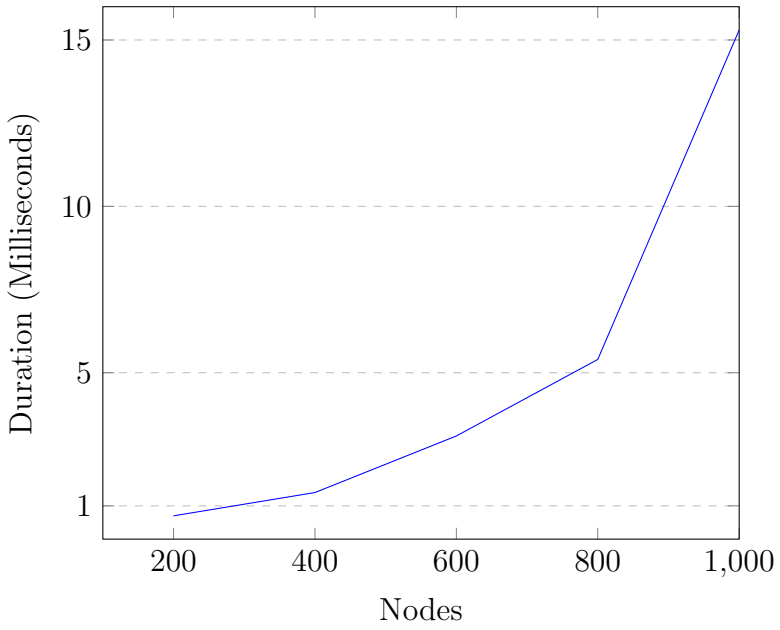

Figure 3.2: DRLEF initialization phase duration

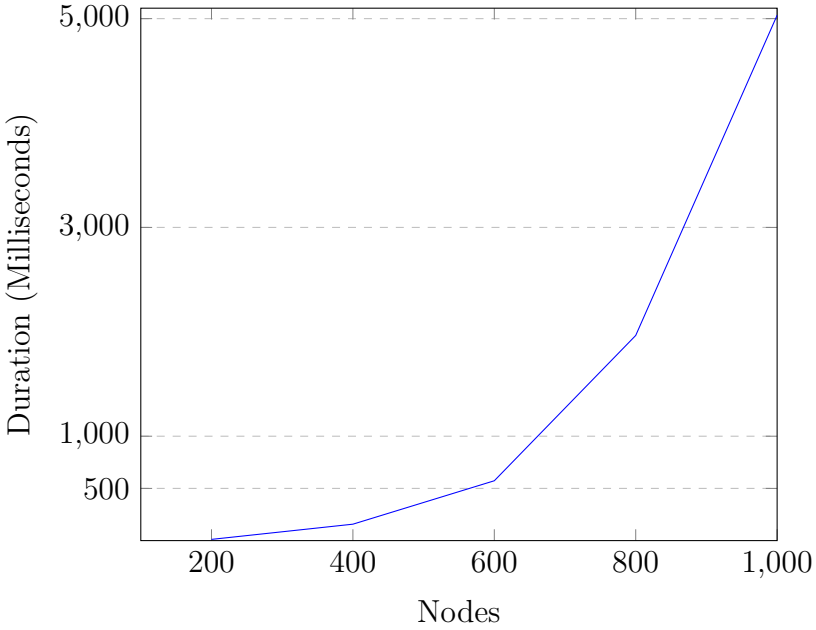

Figure 3.3: DRLEF election phase duration

Fig 3.4 tracing the total time of the protocol, from the start to the end of the election, shows that it remains polynomial. Due to the distributed aspect of the algorithm execution, multiple instances are running simultaneously in different zones of the WSN topology, making it scalable without having a significant impact on the duration of the steps.

Table 3.4 (traced in Figure 3.5) presents a comparison, focusing on the overall duration, between DRLEF and three other leader election algorithms discussed in related works. Time

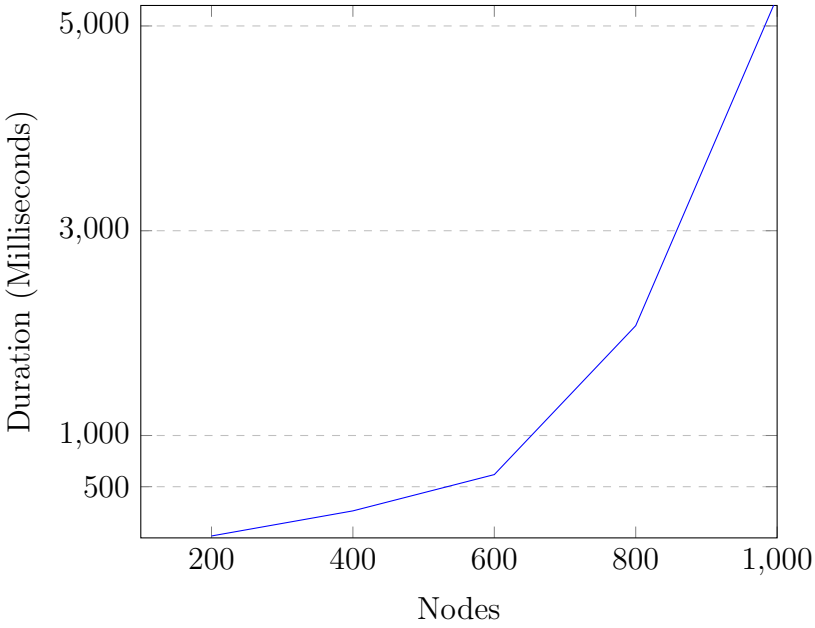

Figure 3.4: DRLEF total running duration

serves as the key criterion since it includes (i) the building of the network map, (ii) the calculation time correlated with the algorithm's complexity, and (iii) the duration of message exchanges, influenced by the number of messages. Given these considerations, we regard time as the central factor for evaluating and comparing algorithms, offering an estimation of other criteria. Based on the comparison, it is evident that our algorithm is less time-consuming, demanding fewer messages and computational operations.

Additionally, it is worth noting that BROGO [57] demonstrates the best performance from a thousand nodes onwards. This can be attributed to its consideration of a flat WSN composed solely of sensors (without gateways), resulting in a complexity close to that of spanning tree algorithms. However, it is crucial to emphasize that this approach may not be suitable for real-world situations and lacks adaptability to diverse scenarios and conditions.

Sensors	200	400	600	800	1000
DRLEF	68,25	153,33	911,83	2054,14	4531,4
MCFATD[66]	1890	3690	5490	7290	9090
ICNP [58]	2000	5000	7500	12000	14000
BROGO [68]	3400	3500	3600	3700	3800

Table 3.4: Leader election algorithms comparison

2.5 Conclusion

This section introduced the design of DRLEF, a lightweight leader election protocol suitable for designating an authentication server within a collaborative group of Gateways. DRLEF

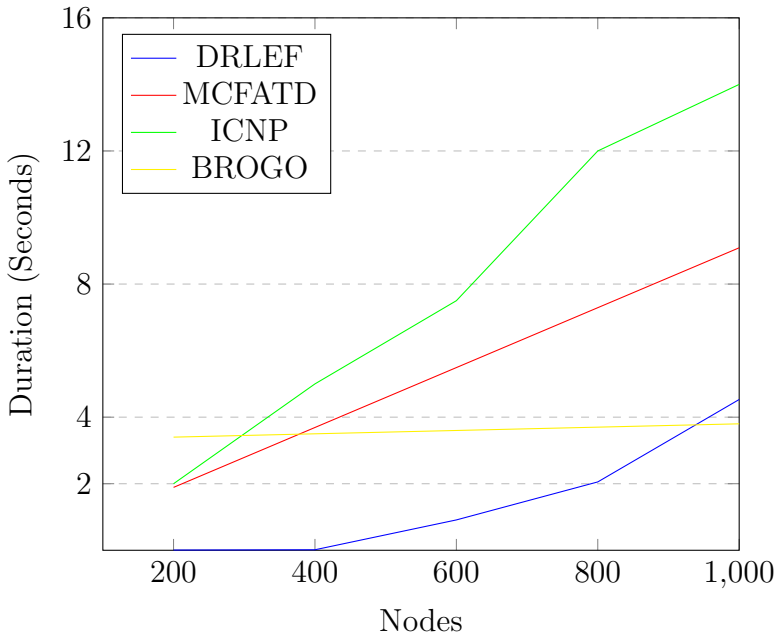

Figure 3.5: Leader election comparative summary

utilizes a logical criterion, centrality, which corresponds to the number of direct competitive neighbors and can be applied in various contexts.

The effectiveness of DRLEF has been demonstrated through simulations, highlighting its scalability with reasonable processing overhead. This efficiency is attributed to the distributed nature of the election procedure. Unlike constructing spanning trees and other structures to cover the entire WSN, we conduct elections locally in each WSN zone. Additionally, we have introduced a failure-tolerance procedure to restart the election when a leader fails.

As a future prospect, we aim to incorporate a procedure to manage mobility. While our algorithm currently supports disconnected or failing nodes, it may face challenges in extreme mobility conditions. Another potential improvement involves enhancing the leader election mechanism by integrating a machine learning-based module.

3 A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment

Cloud computing stands as a pivotal technology in our era, revolutionizing practices for individuals and companies by offering various computing resources as services over the internet. One crucial element contributing to its continued prominence is the assurance of user trust through the maintenance of availability, reliability, and scalability. Upon closer examination of the workings of this technology, it becomes apparent that behind the simplicity of use and service delivery lies a set of highly complex mechanisms working in tandem to ensure optimal functionality [108].

The objective of this work is to introduce a novel hybrid algorithm designed for task scheduling and load balancing in cloud environments [109]. The rationale behind this combination stems from the understanding that effective workload distribution commences with optimizing the task assignment process. Our algorithm encompasses three key stages:

- Servers Clustering (k-means Based): in this initial stage, servers with similar occupation characteristics are partitioned into clusters with limited sizes using a k-means-based procedure.
- Tasks Assignment: this stage involves a two-phase process. Initially, a round-robin algorithm is employed to determine the cluster to which a task will be assigned. Subsequently, a genetic-based algorithm is utilized to select the specific server within this cluster for task scheduling.
- Load Balancing: this phase consists of two steps. The algorithm decides which cluster to unload and precisely which servers to target. Once determined, the cloudlets (virtual machines running them) are retrieved and sent (migration) to the tasks assignment module of the destination cluster for reassignment.

By traversing these stages, our hybrid algorithm addresses the intricacies of tasks scheduling and load balancing in cloud environments.

To achieve this, our architectural model incorporates the following components. Initially, we introduce a new module called the cluster manager, responsible for managing primitives related to the creation and updating of clusters. Subsequently, we utilize a conventional datacenter-level task scheduler, referred to as global, to which we integrate a local monitor within each cluster. Finally, we adopt a similar organizational structure for the load balancer, comprising a central module and local probes deployed in each cluster.

The method we propose relies on realistic assumptions and utilizes specific configurations of existing architectural components. The primary contributions of our work include:

- An algorithm with hot-deployment capability that can be seamlessly integrated into existing operational cloud environments.

- A highly scalable algorithm that maintains consistent performance even with a substantial increase in the number of servers and cloudlets in the datacenters, achieved through the division into clusters and operation at two levels.
- A robust interoperability and complementarity strategy that prevents interference in the mission of each component and redundancy of actions among clusters' management, load balancing, and task scheduling mechanisms. This minimizes non-essential delays associated with cloud management and significantly reduces the number of SLA violations.

The rest of this section is organized as follows: We start by formulating the problem statement with the subsection 3.1, then we put forward a few assumptions in subsection 3.2. The subsection 3.3 introduces our architectural model, we finally depict our method and give corresponding algorithms in 3.4. Results of realized simulation are given and discussed in subsection 3.6. The subsection 3.5 depicts the implementation methodology.

3.1 Problem statement

Considering that the primary objective of a hybrid algorithm for task scheduling and load balancing is to determine the assignment of a task to a specific virtual machine running on a particular server, and subsequently, to address the approach that maintains a workload distribution, eliminating overloaded and under-loaded servers to achieve a balanced and equitable usage level. In this section, we will formalize the problem and introduce the relevant formulas.

Let assume that each datacenter is composed of a set denoted S of N servers$S = \{S_1, S_2, ..., S_n\}$ to which corresponds a set of *Resources* such that:

$$Resources = \{\{R_1^{cpu}(t), R_1^{ram}(t), R_1^{bw}(t), R_1^{str}(t)\}, \{R_2^{cpu}(t), R_2^{ram}(t), R_2^{bw}(t), R_2^{str}(t)\}, ..., \{R_n^{cpu}(t), R_n^{ram}(t), R_n^{bw}(t), R_n^{str}(t)\}\} \tag{3.1}$$

Where $R_i^{resource}(t)$ gives the remaining level of $resource$ of server i at instant t. We assume also that on each server evolve a set VM of M virtual machines such that: $VM = \{vm_1, vm_2, ..., vm_n\}$. Each virtual machine lives on a physical server and has dedicated resources, for example the first vm assigned to the first host noted vm_{11} has $\{R_{11}^{cpu}(t), R_{11}^{ram}(t), R_{11}^{bw}(t), R_{11}^{str}(t)\}$ as remaining resources at instant t. We assume that all virtual machines are initially of a same fixed amount of resources: 2 CPU cores with each 5000 $MIPS$ and 4 Go of RAM.

Each task is characterized by two primary parameters: its length and resource utilization model, often referred to as the cloudlet model. Based on its type, a cloudlet has a specified size in Millions of Instructions Per Second (MIPS) and a variable percentage (ranging from 0.2 to 0.8) of available resource utilization, such as RAM. Table 3.5 provides an overview of available server types with corresponding resources (number of processing cores, RAM, and each core's calculation capacity) and potential cloudlet sizes.

Type	CPU cores	RAM (Go)	Core size (MIPS)	Cloudlet size (MIPS)
Small	2	8	10 000	30 000
Medium	4	16	20 000	50 000
Large	8	32	30 000	70 000
Extra large	16	64	40 000	100 000

Table 3.5: Hosts and cloudlets configuration

To formally articulate the problem, we will introduce key parameters and related equations, categorizing them into two main groups: (i) temporal-based parameters (equations 3.2 to 3.8) and (ii) load-based parameters (equations 3.9 to 3.14).

The equation 3.2 allows the calculation of computation power of a host or a virtual machine:

$$CP_i = |P.U| * sizeof(p.u) \tag{3.2}$$

where $P.U$ is the set of allocated processing units, $|P.U|$ is its cardinality and $sizeof(p.u)$ is the individual capacity of a processing unit in millions of instructions per second (MIPS).

The execution time of a task on a specific virtual machine is given by equation 3.3:

$$ETji = \frac{sizeof(task_j)}{CP_{VM_i}} \tag{3.3}$$

The equation 3.4 gives the full completion time of a task on a virtual machine:

$$CT_{ji} = WT_{ji} + ET_{ji} \tag{3.4}$$

where WT_{ji} is the waiting time of the $task_j$ on the $virtual\_machine_i$ and is given by equation 3.5:

$$WT_{ji} = \sum_{k=1}^{j-1} CT_{ki} \tag{3.5}$$

i.e: waiting time of a task on a specific virtual machine is the sum of completion times of all preceding tasks.

We can now define the makespan, a crucial parameter measuring the overall completion time of all tasks. It is respectively given at the virtual machine, host, and cluster levels by equations 3.6, 3.7, and 3.8:

$$makespan(VM_i) = \sum_{j=1}^{n} CT_{ji} \tag{3.6}$$

$$makespan(host_i) = \max\{makespan(VM_{ji})\} \tag{3.7}$$

$$makespan(cluster_i) = \max\{makespan(host_{ji})\} \tag{3.8}$$

Now, having covered the temporal or time-based parameters, we will proceed to define the most crucial load-related parameters. First of all, the parameter estimating the load on the processing unit is given by equation 3.9:

$$CL_i^t = CL_i^{t-1} + CL(Tasks_{ji}^t) - CL(finished\_tasks_{ji}^t) \tag{3.9}$$

In the same manner we can obtain the load on the RAM and storage at a particular timestamp by the equations 3.10 and 3.11 respectively:

$$RL_i^t = RL_i^{t-1} + RL(Tasks_{ji}^t) - RL(finished\_tasks_{ji}^t) \tag{3.10}$$

$$SL_i^t = SL_i^{t-1} + SL(Tasks_{ji}^t) - SL(finished\_tasks_{ji}^t) \tag{3.11}$$

This estimation is straightforward, as on a particular virtual machine at a specific timestamp, the load corresponds to the already present workload, to which we add the load induced by newly assigned tasks and subtract the load of finished ones. The bandwidth utilization on a virtual machine is obtained by summing the amounts of data streams generated by active tasks:

$$BDU_i^t = \sum_{i=1}^{n} data\_stream(task_i) \tag{3.12}$$

The global load score on a virtual machine is then given by equation 3.13:

$$Load(VM_i) = \alpha * CL_i + \beta * RL_i + \gamma * SL_i + \sigma * BDU_i \tag{3.13}$$

where $\alpha, \beta, \gamma and \sigma$ are pondering and normalizing factors, Then we can calculate the load of a particular server by the equation 3.14:

$$Load(server_i) = \frac{\sum_{i=1}^{m} Load(VM_{ji})}{M} \tag{3.14}$$

These formulas are common and will be utilized in our method to build clusters, define a task scheduling strategy based on a genetic algorithm, and enhance load balancing within a cloud environment.

3.2 Assumptions

To avoid confusion, it is important to establish certain assumptions upon which our approach is built:

- Load balancing is conducted at the cloudlet level; thus, once a virtual machine is created on a specific server, it cannot be migrated to another one.
- A workload is pre-existing in the datacenter before deploying our algorithm, and it is randomly distributed across the servers. An essential advantage of our approach is its capability to facilitate hot deployment in already operational datacenters. Nevertheless, our framework is also applicable to new datacenters without an existing workload.

- We are only interested in the main resources of a server which are: the CPU, RAM, *bandwidth*, *storage*.
- A task is executed continuously without interruption and randomly utilizes virtual machine resources according to a specific model (chosen for the simulation scenario).
- A virtual machine is terminated if it completes the execution of tasks in its queue before the scheduler assigns it new tasks (i.e., there are no fully idle virtual machines).

3.3 Organizational model

Our approach is based on the architectural structure depicted in Figure **??**, which closely resembles standard configurations (refer to Figure 1.3). Combining tasks scheduling and load balancing, our method introduces a critical new module known as the cluster manager, which is central to the focus of this paper. For simplicity, other components such as energy efficiency are deliberately omitted.

We will categorize mechanisms operating at the datacenter level as level-2, and those at the cluster level as level-1. The key components/modules in our solution can be described as follows:

- Requests Handler (RH): Constructs the interaction interface with end users, gathering vital information regarding each request, such as length and deadline. It analyzes, models, and transmits each request to the datacenter broker in the form of a set of tasks.
- Datacenter broker (Broker): is the central module responsible for coordinating all other functional components. It receives information, ensures consistency, synchronizes, and transmits commands to other modules.
- Cluster Manager (CM): is responsible for organizing clusters, meaning it builds and maintains clusters of servers inside a datacenter while relying on the following criteria and primitives:
  - Cluster size: which is a dynamic parameter, but for the first iteration, we randomly fix it to the ratio defined as $N/50$ where N is the total number of servers in the datacenter. It can take any integer value, and experiments are performed to determine its optimal value.
  - Server utilization information: To set up clusters, the module is provided with information on the utilization rates of the resources of each server, mainly: CPU, RAM, $Bandwidth$, *storage*.
  - Fusion and fission primitives: dynamic thresholds are selected to determine when a subset of a cluster should leave it (fission) and create another autonomous cluster. The same mechanism is implemented to determine when and which clusters should fuse to build a larger cluster.
- Cluster monitor: in the assigned cluster, it is responsible for monitoring the evolution of resource utilization on servers. It is recommended to deploy it as a virtual machine in each cluster (similar recommendation for other local components) since clusters are

dynamic and change over time. It continuously collects data from servers, aggregates them into statistical metrics, and transmits decision-making information on which the cluster manager relies to trigger fusion and fission primitives.

- Tasks scheduler: corresponds to the classic module responsible for assigning tasks to the VMs hosted by the servers; however, in our model, it operates on two levels.
  - Level-2 Global Tasks Scheduler (GTS): decides which of the least loaded clusters will receive the incoming tasks.
  - Level-1 Local Tasks Scheduler (LTS): is responsible, within a cluster, for determining which virtual machines will execute the tasks assigned by the GTS.
- Load balancer: is responsible for maintaining a fair workload distribution among servers in the cloud environment. For our purposes, it operates within two levels:
  - Level-2 Global Load Balancer (GLB): decides at datacenter level which cluster (among overloaded ones) should be relieved of workload and to which cluster (among underloaded ones) the virtual machines should be migrated.
  - Level-1 Local Load Balancer (LLB): selects source servers from the origin cluster for the virtual machines migration process. Sink ones within the destination cluster are selected by Local scheduler.

3.4 Algorithm and main phases

Our approach is designed to minimize complexity and reduce delays in task scheduling and load balancing operations. To achieve this objective, it aims to decrease the amount of information and constraints that the task scheduling and load balancing modules need to handle. It begins by decomposing the entire datacenter into clusters, allowing the mechanisms to act on size-reduced sets of hosts. We will first provide an overview of the proposed hybrid algorithm as shown in Figure 3.7and then delve into the details of each step.

3.4.1 Overview

There are two approaches to providing a comprehensive overview of our solution. The first emphasizes the functional model and can be categorized as follows:

1. Clustering: in the initial phase, we employ a clustering procedure based on k-means to categorize servers into four major groups based on their utilization rates and makespans. Subsequently, we further divide these categories into clusters with limited sizes. We introduce primitives called fission and fusion, enabling clusters to evolve in a quasi-cellular manner. Fission allows certain sub-groups of servers to leave a cluster when approaching the centroid of another category. Fusion, on the other hand, facilitates the merging of two clusters if they are in the same category and meet specific size constraints.
2. Tasks scheduling: The second phase focuses on job scheduling, and the module operates at two levels: (i) at the datacenter level, a round-robin procedure determines the cluster to which a group of jobs will be assigned, and (ii) at the cluster level, a genetic algorithm assigns tasks to servers.

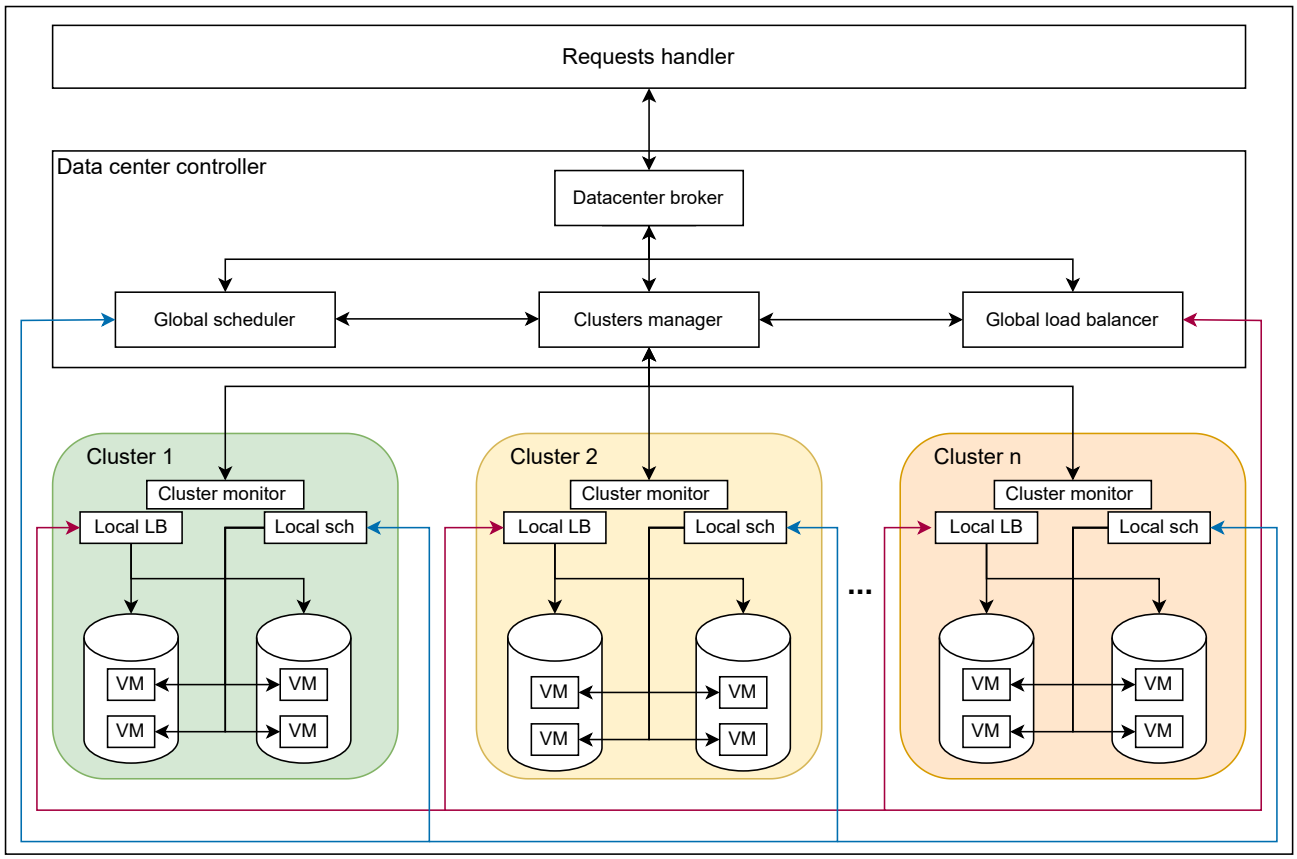

Figure 3.6: Our datacenter organizational model

3. Load balancing: The load balancing stage starts once the tasks are scheduled. Our solution is designed to focus on two tasks: (i) identifying the clusters to be lightened and (ii) locating the servers to be freed. Once this is done, the reallocation of released tasks is handled by the scheduling module.

The second perspective focus on architectural levels on which the mechanisms act and can be depicted as follows:

1. At datacenter level: the mechanisms at this level deal with clusters and are responsible for realizing their respective missions independently while relying on their local modules.

   (a) Cluster manager: is responsible for cluster creation and development. During hot deployment, it initiates the k-means-based clustering procedure. It then supervises cluster evolution by gathering information from local monitors and decides on fission and fusion operations.

   (b) Global tasks scheduler: is in charge of executing a round-robin procedure between clusters to decide which will receive the next jobs. It groups tasks into groups of size equal to the standard number of servers in a cluster, then decides to which cluster to send them.

   (c) Global load balancer: it identifies the overloaded clusters to be relieved by applying

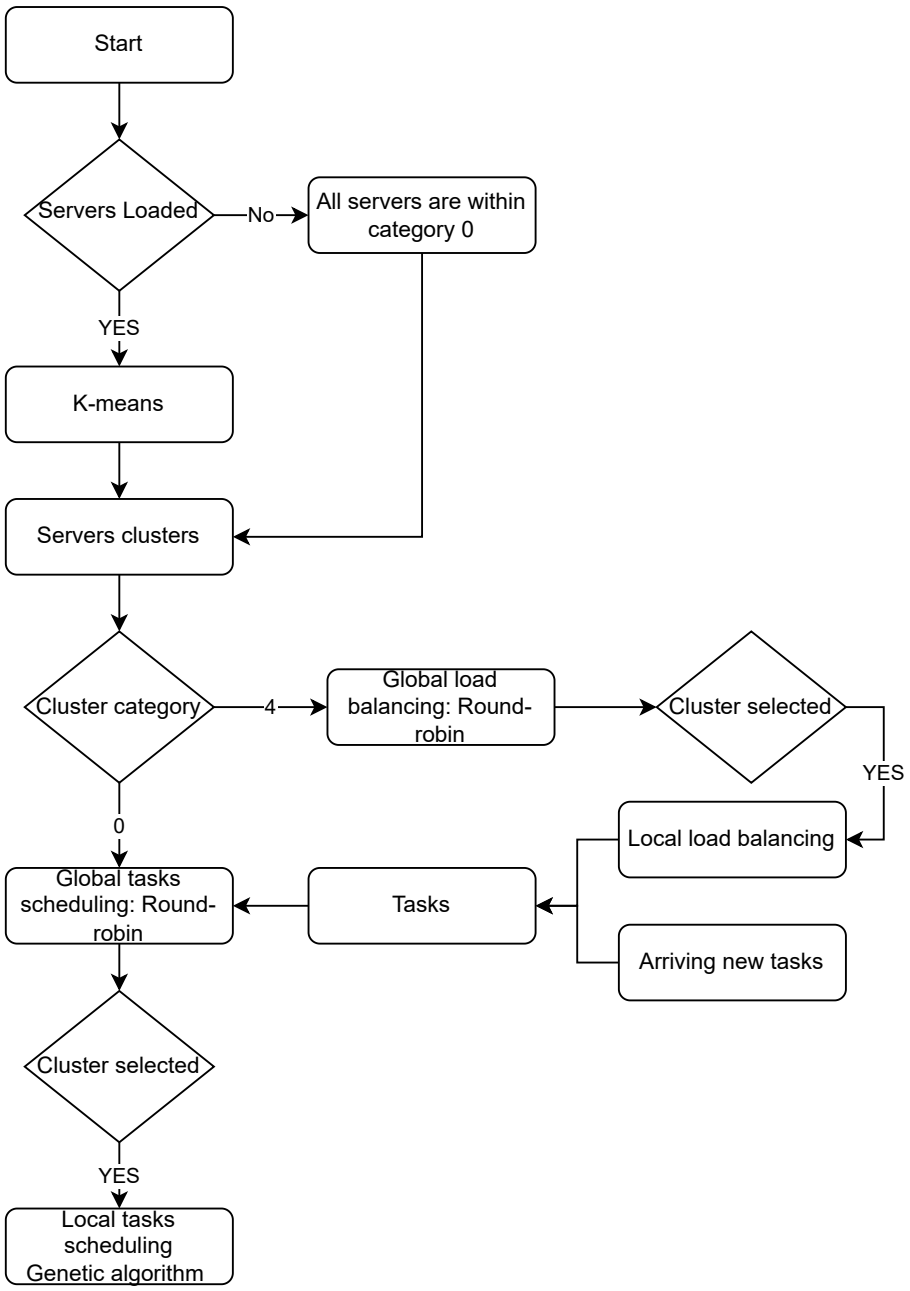

Figure 3.7: Functional model flowchart

a round-robin algorithm between clusters in the fourth category, if any, and those in the third category if none exist.

2. At cluster level: the modules at this level operate within clusters and are responsible for carrying out their tasks on servers, following instructions from global managers and providing them with local information.

   (a) Cluster monitor: keeps track of evolving information within its own cluster, such as server load and cluster size. It continuously monitors the movement of the cluster center and its proximity to the centroids of main categories. It is responsible for

sending alerts regarding the Eulerian distance to the cluster manager when a fission or fusion procedure needs to be triggered.

(b) Local tasks scheduler: receives groups of jobs to schedule from the global module and applies a genetic algorithm to decide on tasks assignment over the servers within the same cluster.

(c) Local load balancer: when summoned by the global load balancer, it executes a specific function to compute a score based on the makespan and utilization rate per server. It then relies on these scores to determine which cloudlets should be retrieved from servers and migrated to another cluster.

This provides a general overview of how the load balancing and task scheduling mechanisms operate based on the architectural organization of the environment. The next subsection delves into the specifics of each step.

First, we'll delve into the construction of clusters at the datacenter level, and subsequently, we'll zoom in on the cluster level, elucidating the fusion and fission primitives.

3.4.2 Clustering

Intuitively, we can conceive a classification of servers according to the criteria of makespan length and scores relative to resources utilization rate, those criteria were previously calculated by the formulas 3.7 and 3.14 and allow to get the following categories as shown in Figure 3.8:

- Category 1: grouping servers with short makespan and low resource utilization rate.
- Category 2: including servers with short makespan and high resource utilization rate.
- Category 3: including servers with long makespan and low resource utilization rate.
- Category 4: grouping servers with long makespan and high resource utilization rate.

Category 1 encompasses under-loaded servers, while category 3 represents a suboptimal group, consisting of poorly utilized servers. Category 4 comprises overloaded servers. The ideal category is category 2, consisting of servers that efficiently use resources, resulting in a reduced makespan. In the load balancing algorithm detailed in 3.4.4, migration operations aim to maximize the number of servers within category 2.

In order to perform this clustering we propose the algorithm 6 which is based on k-means method and can be explained as follows:

- First the algorithm 6 takes as input the list of servers and is expected to return a set of clusters with corresponding servers.
- In order to realize k-means clustering informative features on servers must be modeled. We propose the feature representation given in instructions 2 and 3 respectively using equation 3.7 for the first feature which is the makespan of the server. Then combining equations 3.13 with 3.14 for calculating what we call resource utilization rate score to build the second feature.

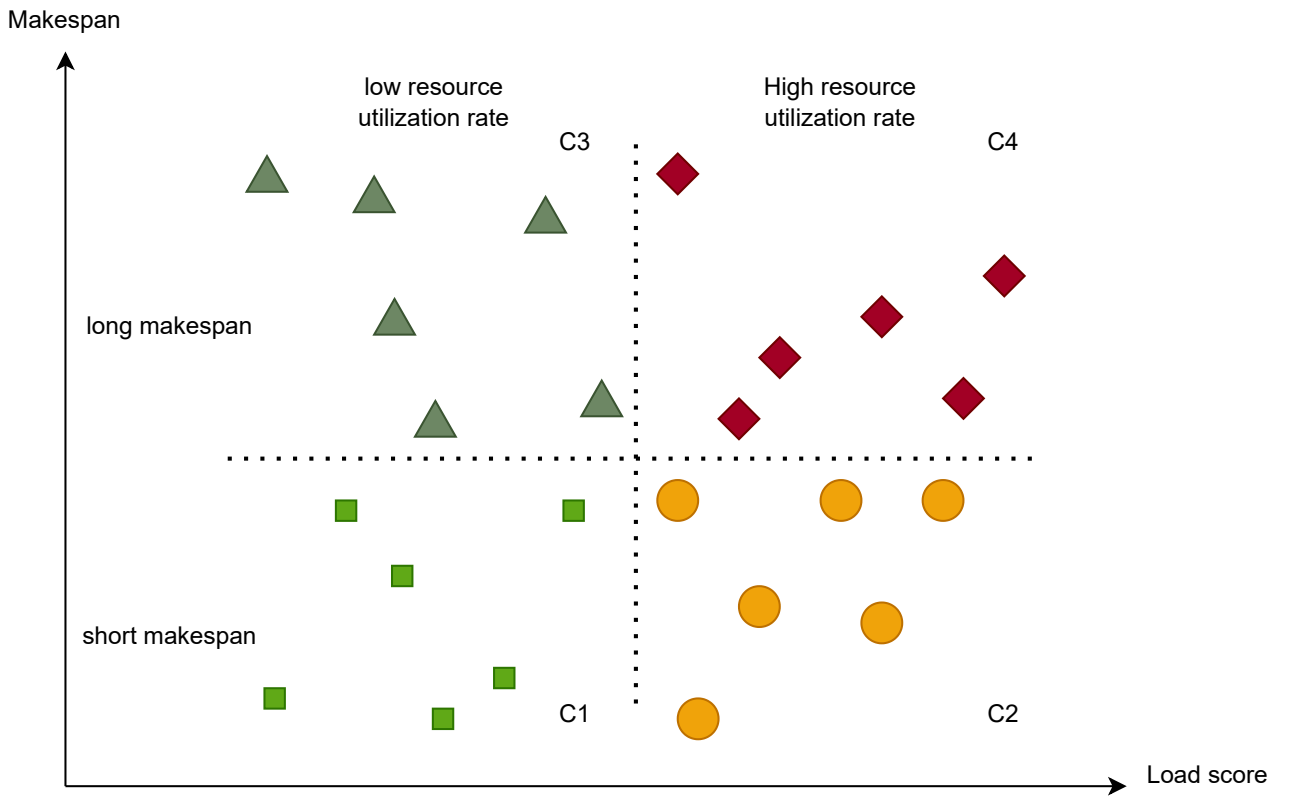

Figure 3.8: Datacenter servers categorization

- Instructions 7 and 8 allow initialization of what we call main centroids coordinates to optimize running time in such a manner that those centroids are initially positioned within the servers features range. Where k is the parameter specifying the number of expected clusters which is four in this case.
- Instruction 11 calls a standard k-means clustering procedure. It takes as argument a list of four main centroids generated randomly which should be updated through several iterations by using euclidean distance and the list of concerned servers. It returns four clusters grouping servers with similar characteristics. This classification around these four main centroids will serve as the basis for our next primitives and algorithms.
- Finally instruction 12 cuts clusters into smaller server pools to render them easier to manage. It takes as input the list of clusters and the desired size which is here equal to 50 and returns a list of clusters of limited size. Local centroids must be recalculated for each cluster and the CM (cluster manager) keeps track of the initial four main centroids.

After creation, the cluster manager installs a cluster monitor as a virtual machine on each cluster. The cluster monitor is tasked with collecting information on servers and relaying it to the cluster manager. The coordinates of the four main centroids obtained from the k-means algorithm are communicated to each cluster monitor, enabling it to execute fusion and fission primitives.

Algorithm 6: K-means servers clustering

Data: List of servers
Result: Clustered servers
foreach $S_i \in Servers$ **do**
  $makespan(S_i) = \max\{makespan(VM_{ji})\}$;
  $utilization\_score(S_i) = \sum(\alpha * CL_{ji} + \beta * RL_{ji} + \gamma * SL_{ji} + \sigma * BDU_{ji})$;
end
$i = 0$;
while $i < k$ **do**
  centroid$[i][X_coordinate]$=$random[min(makespan_i), max(makespan_i)]$;
  centroid$[i][Y_coordinate]$ = $random[min(utilization\_rate\_score_i), max(utilization\_rate\_score_i)]$;
  $i = i + 1$
end
clusters = KMeans_clustering(List centroids, List servers);
split_clusters(clusters, 50);

- Fission: is a function that allows to update clusters so that they stay consistent.
  - When the cluster monitor identifies that one-third of its servers have shifted in proximity, moving closer to one of the three main centroids than to the local centroid, it notifies the cluster manager.
  - The CM initiates a fission operation that produces two clusters.
  - The CM installs a cluster monitor on the new cluster.
  - Obviously, the original cluster updates its local centroid and the new one calculates its own local center.
- Fusion: is a function designed to prevent clusters from becoming overcrowded. The cluster manager monitors the sizes of clusters and, under certain conditions, can initiate the fusion primitive:
  - Two or more clusters are near to the same main centroid.
  - Two or more clusters have number of participant servers under a threshold, let it be 25. Tests can be conducted to evaluate its ideal value which allows a better performance of the method.
  - The cardinality of the union of two or more clusters does not exceed a certain threshold, assuming it is set to 75 for our case.

If the datacenter is devoid of workload at the deployment of this framework, we skip the k-means clustering and initially divide servers into groups of a fixed size.

Now that we have demonstrated how clusters are constructed and established the strategy governing their management, we can proceed to the task assignment and load balancing algorithms.

3.4.3 Tasks assignment and scheduling

After outlining the procedures for cluster creation and management, we will now detail the algorithms for task assignment. It's crucial to note that we have decomposed the datacenter into a set of clusters, enabling us to carry out the required operations at two levels: the datacenter and cluster levels.

At the datacenter level, our attention is directed towards the approach used to assign tasks to clusters. The global scheduler determines which cluster will host incoming tasks, irrespective of how they will be managed locally by the local task scheduler module. To achieve this, the global scheduler operates as follows:

- First, it selects the targeted category of clusters. It naturally prioritizes clusters of category 1, given that they contain under-loaded servers. If no cluster is available in this category, the global task scheduler (GTS) explores possibilities in class 3. The optimistic approach is taken here, as this category underutilizes its resources, and there is a chance to enhance this ratio without significantly impacting the makespan. As a last resort, it will opt for class 2, representing the ideal exploitation model that should not be disturbed.
- Then, it determines the targeted cluster using a round-robin algorithm. Once the category is selected, the global scheduler compiles a list of all corresponding clusters and subsequently redirects the received tasks to these clusters in a sequential manner. Based on a given parameter, representing a specific number of tasks, after scheduling this quantity on the clusters, it repeats the process by revisiting the initial step.

We suggest commencing with the definition of an algorithm to locate clusters corresponding to a specific category in relation to the four main centroids. Algorithm 7 elucidates the process.

Algorithm 7: Find clusters

```
Data: List of clusters, category
Result: All clusters of specific category
found = Null;
i = 0;
foreach cluster ∈ clusters do
    if cluster.category = category then
        found[i] = cluster.id;
        i = i + 1;
    end
end
return found;
```

The algorithm 8 relies on the algorithm 7 and allows to select the clusters that will be concerned by round-robin tasks assignment procedure.

Algorithm 8: Select clusters

Data: List of clusters
Result: Selected clusters for tasks assignment
$selected\_clusters = find\_clusters(clusters, 1)$;
if $selected\_clusters = NULL$ **then**
  $selected\_clusters = find\_clusters(clusters, 3)$;
end
if $selected\_clusters = NULL$ **then**
  $selected\_clusters = find\_clusters(clusters, 2)$;
end
if $selected\_clusters = NULL$ **then**
  $selected\_clusters = find\_clusters(clusters, 4)$;
end
return selected_clusters;

An essential procedure for the round-robin algorithm needs to be outlined. Algorithm 9 receives a list of selected clusters and a single task, assigns the task to one of these clusters, and outputs the remaining list.

Algorithm 9: Assign task

Data: List of clusters, one task
Result: task assigned to one cluster
$cluster = random(clusters[0..Length])$;
$schedule(task, cluster)$;
$clusters = clusters - \{cluster\}$;
return $clusters$;

Now we have defined all necessary functions we can introduce the round robin method as shown in algorithm 10.

Algorithm 10: Round robin tasks assignment among selected clusters

Data: List of clusters, List of tasks
Result: tasks assigned to clusters
$selected\_clusters = select\_clusters(clusters)$;
$task = random[0..Length]$;
while $not\_empty(clusters)$ *or* $not\_empty(tasks)$ **do**
  $selected\_clusters = assign(selected\_clusters, task)$;
  $tasks = tasks - task$;
end

Element	Description
Coding	- Gene: binary value expressing if the ith server is in charge of a task. - Chromosome / Individual: Vector of N genes where N is the number of servers in the cluster. Each expresses a particular disposition which is a feasible solution - Population: a set of individuals
Fitness function	Fitness= 1 - $\frac{CUR+CMS}{2}$
Parents selection	Elitist, tournament
Crossover	Uniform crossover
Mutation	Randomly made on each gene according to a fixed mutation rate threshold

Table 3.6: Genetic algorithm based local scheduler parameters

The local task scheduler functions at the cluster level and does not impact the global scheduler policy. For local task scheduling, the relevant module utilizes a genetic algorithm. Configuration elements of the algorithm are provided in Table 3.6, where CUR represents cluster utilization rate, and CMS represents cluster makespan generated (added) by a particular solution and can be obtained by equations 3.15 and 3.16:

$$CUR = \frac{\sum_1^N UR_i}{N} \tag{3.15}$$

$$CMS = \frac{\sum_1^N makespan_i}{N * \max\{makespan_i\}} \tag{3.16}$$

To perform task scheduling at the cluster level, we chose to employ a variant of genetic algorithms due to their suitability for the type of problem and the conclusive results obtained by works based on them. To design a robust and efficient genetic algorithm, we need to carefully consider three elements: (i) the generation of individuals and the population, (ii) the operators for the evolution of populations, and (iii) the procedure for executing these operations on individuals based on the fitness function.

For the initial generation, we need to create a random population of feasible solutions. This is achieved by generating one feasible solution at a time using Algorithm 11. This algorithm incorporates constraints such as the number of available servers and arriving tasks, producing a random realizable solution.

Then we repeat the procedure for individual creation a certain desired number of times to create the first generation population, this is explained in algorithm 12.

Algorithm 11: create an individual

```
Data: Number of tasks, number of servers
Result: A feasible solution
individual = [0, 0, ..., 0]
for i = 0; i < servers.length; i++ do
    while count < tasks.length do
        gene = random{0, 1};
        individual[i] = gene;
        count = count + 1;
    end
end
return individual;
```

Algorithm 12: Generating population algorithm

```
Data: number of individuals
Result: Population of feasible solutions
for i = 0; i < desired_number; i++ do
    population[i] = create_individual(M, N);
end
return population;
```

After generating a random set of individuals to build the first generation of solutions we move on now to introduce the operators handling theses chromosomes.

The selection operator, as outlined in Algorithm 13, is crucial for choosing the most suitable individuals to act as parents for reproduction. We have employed a tournament selection method, offering the advantages of speed, diversity, and giving less optimal individuals a chance to participate with better chromosomes in the creation of the new generation. The method randomly selects a certain number of individuals from the entire population and pits them against each other to determine the best one.

Algorithm 13: Tournament selection operator

```
Data: Population, number of participants
Result: Best individual (fittest)
best = NULL;
for i = 0; i < number_of_participants; i++ do
    individual = random(population[0..Length]);
    population = population − individual;
    if individual.fitness() > best.fitness() or best = NULL then
        best = individual;
    end
    return best;
end
```

The first evolutionary operator is the crossover operator, outlined in Algorithm 14. This operator is evolutionary in the sense that it creates a new generation from the original population, and the offspring is expected to consist of better solutions than those in the parent generation. We chose to use the uniform crossover approach for this operator, where each gene of the offspring is inherited from either parent 1 or 2 based on a uniformly distributed probability.

Algorithm 14: Crossover evolution operator

```
   Data: Population
   Result: New generation
1  individuals = NULL;
2  while not_empty(population) do
3  |  parents[0] = tournament_selection(population);
4  |  parents[1] = tournament_selection(population);
5  |  for i = 0; i < parents[0].length; i++ do
6  |  |  choice = random{0, 1};
7  |  |  if choice = 0 then
8  |  |  |  genes[i] = parents[0][i];
9  |  |  end
10 |  |  else
11 |  |  |  genes[i] = parents[1][i];
12 |  |  end
13 |  |  individuals[j] = genes;
14 |  |  population = population − parents;
15 |  |  j = j + 1;
16 |  end
17 end
18 return individuals
```

Now that all the elements that go into the operation of a genetic algorithm are assembled, the overall orchestration is performed by the algorithm 15.

Algorithm 15: Genetic algorithm for in-cluster tasks scheduling

Data: List of ordered tasks, lists of servers, number of iterations
Result: Best solution as execution planning on servers

```
populations[0] = generate_population();
for i = 0; i < number_of_iterations; i++ do
    foreach individual ∈ populations[i] do
        fitness[j] = individual.fitness();
        j = j + 1;
    end
    populations[i + 1] = crossover(population[i]);
    foreach individual ∈ populations[i + 1] do
        individual = mutate[individual];
    end
end
fittest = NULL;
foreach individual ∈ population[i] do
    if individual.fitness() > fittest.fitness() or fittest = NULL then
        fittest = individual;
    end
end
return fittest;
```

3.4.4 Load balancing

The effectiveness of our approach relies on two key elements. First, clustering enables the management mechanisms to operate on two levels, allowing us to handle a reduced number of entities—whether clusters at the datacenter level or servers at the cluster level. The second key element is the high degree of decoupling in the missions of the various modules. As mentioned earlier, during the load balancing step, our focus is on a single question at two levels: determining which clusters to relieve and identifying the specific servers to be lightened. This is because the load balancer is no longer responsible for migrating cloudlets; instead, the scheduler will reassign them to other servers in category one clusters.

1. At datacenter level: the initial stage of the load balancing process involves identifying clusters within the fourth category—those characterized by high resource utilization rates and long makespans. If category four clusters are available, the algorithm attempts to locate clusters in category three that exhibit suboptimal utilization rates but still have a long makespan. This is accomplished through the use of Algorithm 16.

   Once found, the clusters of category four or ones of category three if the fourth one is empty should be lighten. A round-robin algorithm described in 17 is again used at datacenter level to ensure fairness between clusters.

2. At cluster level: Upon receiving instructions from the global load balancer, the local load balancer initiates the process of determining the servers to be freed and the cloudlets to be migrated to another cluster. To achieve this goal, the load balancer begins by calculating

Algorithm 16: Select clusters to unload

```
Data: List of clusters
Result: Selected clusters to unload
selected_clusters = find_clusters(clusters, 4);
if selected_clusters = NULL then
    selected_clusters = find_clusters(clusters, 3);
end
return selected_clusters;
```

Algorithm 17: Round robin to relieve selected clusters

```
Data: List of clusters
Result: Cluster to relieve
selected_clusters = select_clusters(clusters);
i = 0;
while not_empty(clusters) do
    relieved_cluster = relieve(selected_clusters[i]);
    selected_clusters = selected_clusters − relieved_cluster;
    i = i + 1;
end
```

the mean makespan of servers in the cluster, denoted as MMS, as determined by Equation 3.17:

$$MMS_{C_i} = Mean(Makespan(server_{ij})) \tag{3.17}$$

The final step in our method is executed using Algorithm 18, which the local load balancer employs to determine the cloudlets that need to be relieved from servers and migrated to another cluster. It's important to note that the task of migrating these cloudlets is handled by the global tasks scheduler, and it falls outside the scope of this algorithm.

Algorithm 18: Determine cloudlets to migrate

```
Data: List of cloudlets within the cluster
Result: List of cloudlets to migrate
foreach cloudlet ∈ cloudlets do
    Planned_time = cloudlet.getPlannedOnTime();
    if Planned_time > MMS then
        migration_list = migration_list + cloudlet;
    end
end
return migration_list;
```

Now that we have provided a detailed explanation of our method, including each step, the roles of individual architectural components, and the strategies employed to trigger primitives, we will proceed to the next section to elucidate our validation method. We will delve into the implementation details, discuss the results obtained, and compare them to the best findings in the existing literature.

3.5 Implementation

In order to validate our method, we implemented it using the standard cloud simulator called CloudSim Plus. Table 3.7 outlines the necessary implementations in terms of objects and their corresponding parameters.

CloudSim is an open-source simulation library extensively used for modeling and evaluating cloud computing systems. It provides researchers and developers with the capability to simulate cloud datacenters and applications, allowing them to assess performance metrics such as energy consumption, makespan, and more. CloudSim offers a simple and flexible interface for creating customized simulation scenarios, enabling researchers to focus on implementing their algorithms, such as virtual machine placement, tasks scheduling, and load balancing. These algorithms can then be evaluated based on specific criteria, such as the number of virtual machine migrations [110]. CloudSim Plus serves as an extension to the CloudSim simulation library, introducing advanced features and performance enhancements to expedite the modeling and simulation of cloud computing environments [111].

Simulation	Datacenter	Hosts	Virtual machine	Cloudlets
Datacenter	List of hosts	CPU	CPU	Length (MIPS)
Broker	VM allocation policy	RAM	RAM	Utilization model
List of hosts		Bandwidth	Bandwidth	
List of VMs		Storage	Storage	
List of cloudlets		Resource provisioner	Cloudlet scheduler	

Table 3.7: Cloudsim simulation model and elements

The simulation was conducted on a laptop with following characteristics:

- Processor: Intel ®Core ™i7-10510U CPU @ 1.80GHz.
- RAM: 16 GO.
- OS: Windows 11 64 bit, x64-based processor.

To validate the accuracy of our results, we executed our algorithms in multiple scenarios, manipulating crucial parameters such as the number of servers, cloudlets, and cluster sizes. We gathered essential metrics, including the duration across different stages and the number of operations. Subsequently, we conducted a comparative and analytical study, comparing our method against those considered most relevant in the existing literature.

The gathered metrics are derived from calculating mean values across one hundred repetitions for each scenario. Cloudlets and servers are generated with random parameters, as detailed in Table 3.5, to enhance the realism of the scenarios.

3.6 Results discussion

Table 3.8 presents the primary performance metrics used to assess the efficiency of our method and to facilitate comparisons with other approaches. Given the negligible time involved in load balancing processes, our focus is on three key metrics: response time, the number of migrations, and the number of SLA violations.

Parameter	Description
Duration	• Clusters management duration: – Clustering – Primitives: fission and fusion • Tasks scheduling times: – Round-robin among clusters – Genetic algorithm inside a cluster
Migrations	Number of cloudlets selected by local load balancer for migration
SLA violations	Number of cloudlets violating the service-level agreement
Makespan	Obtained by equation 3.7
Response time	Refers to the time required by the load balancer to detect an unbalanced situation and to determine cloudlets to migrate

Table 3.8: Load balancing evaluation metrics

We calculated the clustering time based on the number of servers within a datacenter and for varying cluster sizes. The results are presented in Table 3.9. In comparison, the approach proposed by [112] takes 4 seconds for clustering one thousand servers. With the parameters used in the k-means algorithm, our approach allows for clustering operations within a very minimal time frame.

Number of servers	2000				5000				10000				20000			
Cluster size	50	100	200	500	50	100	200	500	50	100	200	500	50	100	200	500
Clustering duration	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.03	0.04	0.04	0.05	0.05	0.05	0.05

Table 3.9: K-means clustering duration

Evaluating task scheduling involves considering two durations: (i) the time required for the global scheduler to process group tasks in batches and execute round-robin to designate the target cluster, and (ii) the time required for the local scheduler to execute the genetic algorithm and assign tasks to servers. The detailed results are presented in Table 3.10. When focusing on overall times, it becomes evident that smaller clusters yield better performance. This is attributed to the negligible round-robin time, even with a large number of clusters, and the substantial increase in runtime of the genetic algorithm as the number of servers grows.

Number of servers	5000				10000			
Cluster size	50	100	200	500	50	100	200	500
Round-robin duration	0.002	0.002	0.002	0.002	0.005	0.005	0.005	0.005
Genetic algorithm duration	0.2	0.31	0.68	3.19	0.47	0.46	2.39	3.5
Total scheduling time	0.202	0.312	0.682	3.192	0.475	0.465	2.395	3.505

Table 3.10: Tasks scheduling duration

The results from Table 3.10 are visualized in the graphs presented in Figure 3.9. The graphs reveal a degree of irregularity, as the regression is not entirely linear between cluster sizes and scheduling times. This observed phenomenon is attributed to the random generation of task characteristics and server capacities. The results are obtained by aggregating the outputs of various scenario repetitions into average values.

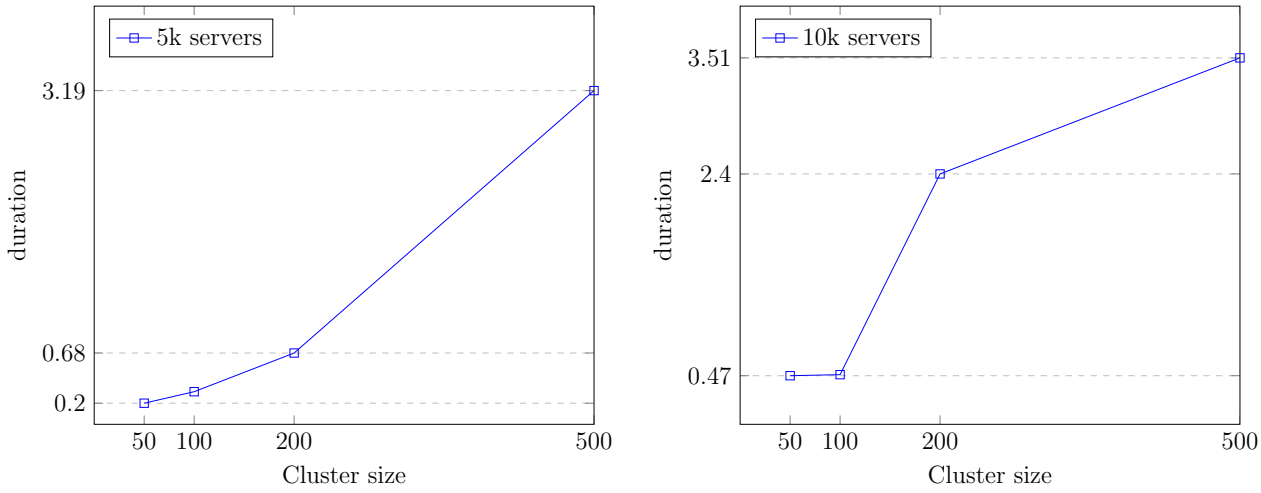

Figure 3.9: NHL-BA2C tasks scheduling duration according to cluster size

Table 3.11 provides a comprehensive overview of the key performance metrics for our approach across various cluster sizes. The main observations can be summarized as follows:

- The number of cloudlets to migrate logically increases with the cluster size. This is attributed to the size of the task batches transferred by the global scheduler to the cluster, which equals the number of servers within the cluster. As the cluster receives batches of tasks with varying sizes, the load balancer decides to migrate a higher number of cloudlets

to prevent potential SLA violations. It's noteworthy that the proportion of cloudlets to migrate remains close to the ratio of 12% relative to the cluster size (batch of tasks).

- Regarding the given makespan, it pertains to a single batch of tasks with random sizes. The observed makespan falls within a notably favorable range of values. Larger clusters may exhibit a longer makespan due to the increased diversity in the characteristics of the cloudlets they receive.

- The number of SLA violations tends to increase with larger clusters. With a higher number of tasks received, the probability of some tasks violating the SLA also increases. Nevertheless, it is essential to highlight that these violations remain below an acceptable threshold, approximately 8%.

- The last row of Table presents the results of the load balancing module's response time evaluation. This duration encompasses the process of identifying the cluster to be lightened and specifying the servers to be relieved. Essentially, it includes the time from detecting an imbalance to determining the list of cloudlets to migrate. Notably, the observed delays are negligible. In comparison, the solution proposed in [86] requires $0.008 seconds$ to determine the tasks to migrate among a total of just 30.

Cluster size	50	100	200	500
Migrations	5	13	20	76
Makespan	1.35	1.2	1.5	1.7
SLA violations	5	8	18	35
Response time	0	0.004	0.005	0.009

Table 3.11: NHL-BA2C Global performance evaluation

Now, let's proceed to the comparative analysis. We will begin by comparing the key metrics obtained from our approach with those of other relevant hybrid approaches in the literature. Table 3.12 provides a side-by-side comparison between our method and selected alternatives, using standard parameters. The results were evaluated for a datacenter with 2000 servers and 2000 cloudlets. Notably, our method demonstrates a superior average makespan, a lower migration ratio, and a significantly reduced number of SLA violations.

	Makespan	migrations	SLA violations
Our method	1.18	12%	8%
[88]	3.5	19%	unknown
[80]	8	35%	unknown
[85]	3.8	unknown	18%

Table 3.12: Load balancing techniques comparative analysis

Figure 3.10 visualizes the results presented in Table 3.12. The left plot provides a visual comparison of our method with those presented in [88] and [80] based on the number of performed cloudlet migrations. Meanwhile, the right graph compares our method with [85] in terms of SLA violations. The red lines are included as reference points.

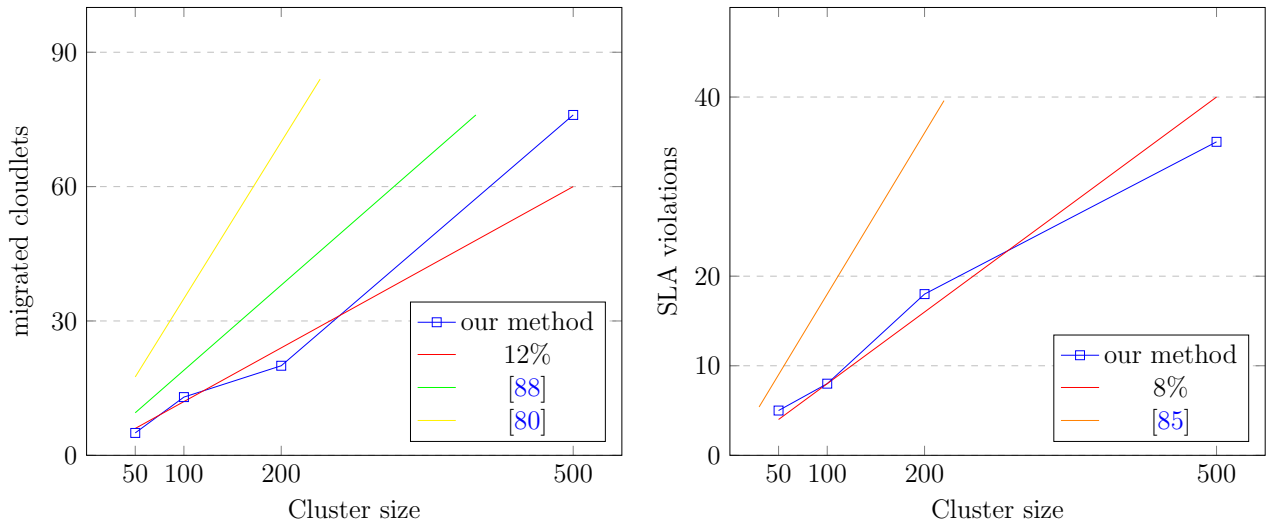

Figure 3.10: NHL-BA2C migrations and SLA violations

The performance evaluation of our method has yielded promising results, showcasing excellent scalability and a reduction in delays, migrations, and SLA violations.

3.7 Conclusion

In this study, we introduced a novel hybrid approach for job scheduling and load balancing in cloud environments. This approach presents several advantages, including hot-deployability, high scalability, decoupling, and robust interoperability among the components managing the cloud ecosystem. Its effectiveness is rooted in its operational method. Initially, servers are clustered using a k-means algorithm, considering criteria such as utilization rate and makespan. This allows us to address challenges on two levels: at the datacenter level and within each cluster. The tasks scheduler comprises two modules—one global and utilizing round-robin to distribute task batches to specific clusters, and the other local, employing a genetic algorithm for task assignment to servers. The load balancer also operates on both levels: a global module using round-robin to identify the cluster for load reduction, and a specific algorithm employing individual scores to determine the servers for unloading. The clustering mechanism includes probes within each cluster, predicting their evolution to execute fission and fusion actions aimed at maintaining cluster coherence.

We successfully validated the effectiveness of our approach by implementing it with CloudSim Plus, yielding highly conclusive results in terms of makespan, response times, service-level agreement (SLA) violations, and cloudlet migrations. The comparative study demonstrated a significant improvement over recent and relevant works in the literature. Moving forward, we aim to

explore future enhancements, leveraging the capabilities of machine and deep learning to optimize cloudlet migration processes. Additionally, we intend to define more optimal thresholds for controlling the actions of our algorithms.

4 A novel privacy-aware global infrastructure for ecological footprint calculator (ANPA-GIEFC) based on Internet of Things and blockchain

In this work, we introduce a novel global infrastructure and the corresponding functional framework for collecting data on activities involving ecological footprints without the need for declarations, new physical devices, or the invention of new technological concepts [8]. The only assumptions required are: (i) each entity must have at least a device connected to the internet, (ii) the existence of trusted authorities on the cloud, namely: first, a Certification Authority (CA) for identities and certificates management, secondly, an authority for registering production and consumption of goods and services, let's call it Prodction/ Consumption Autority (PCA), and finally, a Scoring Authority (Scoring Autority (SA)). The main steps can be depicted as follows:

1. Each entity must register with the Certification Authority (CA), which issues a digital certificate containing information about the owner: a unique identifier, its public key, and its role (producer or consumer). Each entity, having a network of connected objects, must then elect a representative for future actions in the blockchain.

2. Each entity with a producer role must register each new good or service with the Production/Consumption Authority (PCA) using its public certificate. The PCA then issues an attribute certificate, which is embedded in the device or the service platform. The certificate contains information such as a unique ID, producer ID, a precise category of goods/services, an attached unitary ecological impact, and more. The PCA transmits the certificate to the Sustainability Authority (SA), which updates the impact score of the producer in a global table. Subsequently, the SA pushes the certificate to the blockchain to be stored in the distributed ledger.

3. When a customer purchases a good or service and wishes to activate it, they need to subscribe to the PCA. Subsequently, the ID of the consumer is added to the attribute certificate. The PCA then transfers the certificate to the Sustainability Authority (SA), which updates the customer's score and pushes the new certificate to the blockchain for storage as a transaction.

As best as we know this is the first time a such approach is proposed, the elements which make it realist and its main contributions can be summarized in the following points:

- A global behavioural approach to track activities with ecological impacts.
- The approach is preserving privacy and confidentiality since:
  - It relies on public key identification.
  - It saves activities in a permissioned distributed ledger using the public identifiers.

  - Only authorized authorities can assemble tracks and decode the real identity of each entity.

- It provides a real and precise impact score since it does not rely on declarative but behavioural data.
- It is easily achievable as it does not require additional assumptions or devices acquisition.
- It works independently of payment method, it does not require on crypto-currencies or particular method.

Validating such an architecture is a challenging task as it commonly relies on implementation and non-standardized metrics for performance evaluation. In this work, we propose a two-step method to provide evidence of the correct operation of our system:

- Implementation involves using the Python language and related packages for discrete event system simulation. The aim of this step is to provide a runtime step-by-step process that will aid in modeling in the next step.
- Modeling involves two formal methods: (i) Colored Petri networks to demonstrate that our system correctly provides the expected services starting from an initial deployment state. (ii) Queue modeling to verify the stability of our system over time.

The results obtained are positive, indicating that the objectives have been achieved, and the architecture is evolving towards a stable state.

The rest of this section is organized as follows: assumptions are made in 4.1, architectural components are depicted in 4.2, main phases and steps are described in subsection 4.3. In order to validate the ANPA-GIEFC model we propose in subsection 4.4 a validation methodology in three steps, we discuss its performance metrics in the subsection 4.5.

4.1 Assumptions

Some assumptions are mandatory to make our contribution feasible, those are:

- There is a set of trusted certification authorities, and the number is proportional to the volume of activities, and consequently, to the number of network users.
- A user, whether a producer or a consumer, uses a device such as a smartphone or a computer equipped with the capability to interact with a web interface for their initial registration.
- DNS propagation times are ignored because they are out of the scope ou this research proposal.

4.2 Architectural and functional components

In the preceding section, we briefly introduced the fundamental elements required to construct ANPA-GIEFC and provided an overview of existing ecological footprint digital calculators. In this section, we will present our solution, its architectural components, functional elements, and model it with flowcharts. An overview is depicted in Figure 3.11.

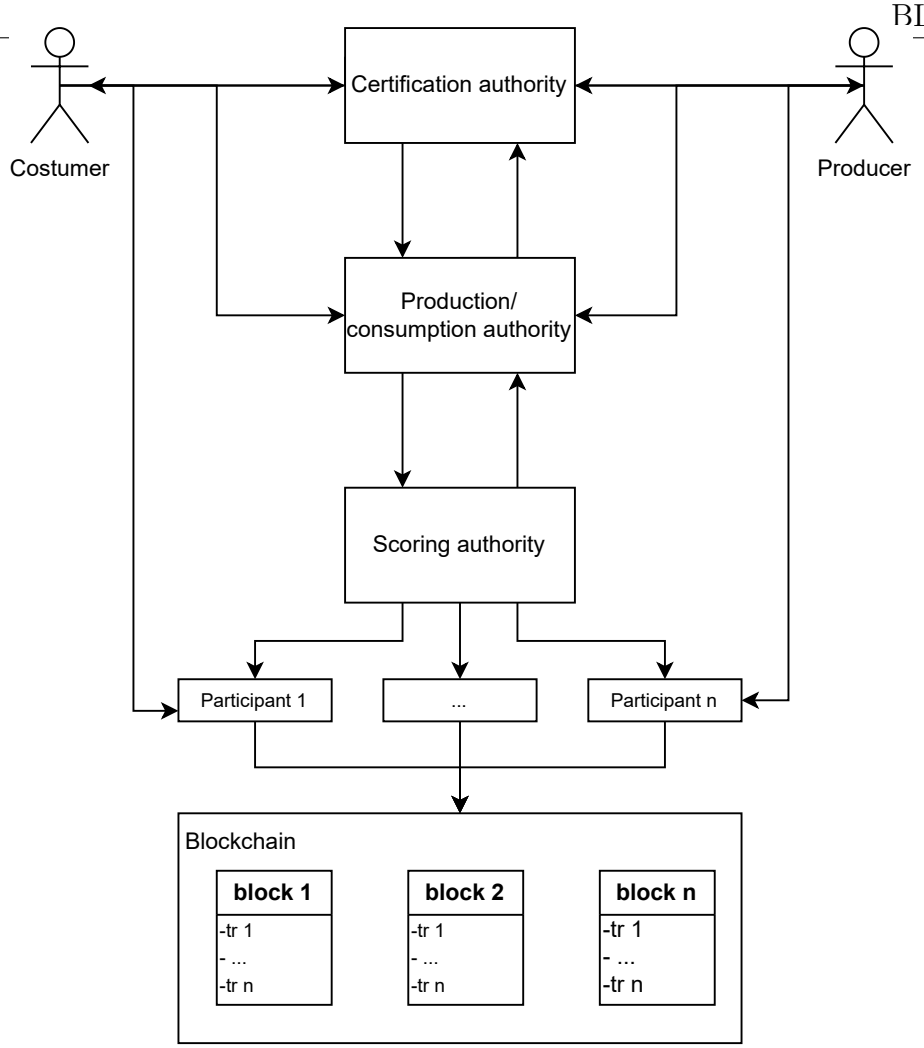

Figure 3.11: Global architecture of ANPA-GIEFC

4.2.1 Architectural components

We will first depict architectural components which are main servers or authorities that form core structure of our model:

1. Certification authority (CA): This is a fundamental element for establishing trust in the network. All identification and access control will rely on it, and its main functions can be summarized in the following primitives:

   - Registering: When a new user (customer or producer) joins the network, they need to register with the Certification Authority (CA), which collects the necessary information, validates their identity and public key, and then generates a dedicated digital certificate for them.
   - Verification: When a transaction is pending, concerned stakeholders can choose to send a request to the Certification Authority to verify the integrity of each other's digital certificates.

- Revocation: When a user is banned, or when their key is leaked or compromised, the respective CA should be able to revoke it. This ensures that subsequent verification requests from participants will receive negative responses.

2. Production/ consumption authority (PCA): it is proposed to assign complementary tasks to the CA. Instead of managing users (producers or consumers) and their identities, it processes the operations between them concerning products and services. It offers the following primitives:

   - Registration of production of new goods or services: when a producer proposes a new service or good for sale, they have to register it within the PCA, which will deliver an attribute certificate containing information about the product or service, the producer, and the ecological impact category.
   - Registration of consumption of a good or a service: when a customer buys a product or uses a service, in order to activate it, they have to register it within the PCA, which will update the information of the attribute certificate by adding the consumer's identifier to it.
   - Deletion of a good or service: when a service or product is obsolete or no longer usable PCA will revoke its attribute certificate.

3. Scoring authority: The functions ensured by this authority are the main purpose of this model; it maintains a global table of identifiers and corresponding ecological impact scores on one hand, and pushes transactions in the form of final attribute certificates to the underlying blockchain.

To completely separate the missions and ensure objectives of transparency and privacy, the model assigns isolated tasks to different authorities. To enable cooperation, a communication model is proposed in this section. Summarily, the functions described here are as follows: (i) CA manages identities, (ii) PCA is in charge of tracking production and consumption activities, (iii) SA calculates ecological impacts for each user, registers it in a dedicated table, and pushes each transaction to the blockchain.

4.2.2 Functional components

After we have defined main architectural components we will now discuss functional elements and justify our choice of each brick.

In order to collect data on activities we decided to choose Internet of Things based approach because of some key features which can be summarized as:

1. Ubiquity and mobility: to enhance accessibility for users, the ideal solution is to provide interfaces that allow them to perform tasks at any time and from anywhere. Instead of considering new terminal equipment or alternative solutions, it is preferable to leverage ubiquitous technology that already offers high mobility. This approach ensures fluid and familiar interactions for users.

2. Integrated sensor layer: in addition to ubiquity, devices within the Internet of Things, such as smartphones, can incorporate sensors to rapidly collect user information, including geolocation. This not only enhances convenience but also strengthens security by integrating features like biometric readers for authentication before conducting purchase operations.

3. Ad-hoc and personal area networks technologies: another useful characteristic of the Internet of Things is the hybridization of accessibility modes: both direct access via the global network and organization in personal area networks. In other words, devices around a user are organized to stay interconnected in ad-hoc mode. Simultaneously, one or more of their devices are connected to the internet and act as relays. This ensures that each user is represented on the network by at least one device, allowing them to participate in blockchain activities.

4. Existence of security approaches and communication protocols: with the Internet of Things (IoT), we discovered lightweight communication protocols that do not overload bandwidth or computational units on one hand, and security mechanisms that have been adapted and improved by researchers on the other hand. This provides a fertile ground to deploy our architecture without concerns about these two aspects.

We chose the distributed ledger option for data storage by progressively evaluating our needs and aligning them with the flowcharts proposed by VIKAS et al. [113] for determining the suitability of using blockchain technology. The flowchart in Figure 3.12 outlines the controls and corresponding responses (dotted arrows) that led us to this decision. The reasons for our choice can be explained as follows:

1. The need for a shared database and the presence of multiple writers to the database led us to opt for the distribution of the database. This ensures a high level of transparency, allowing all users to access data on activities with ecological impact. By distributing the database, each participant can have a copy of at least one fork of the distributed ledger. This approach requires active participation from entities across the network, both in registering new transactions and preserving integrity. Additionally, it aligns with our goal of accommodating various payment methods. Traditional payments can be processed through central servers connected to the PCA, while cryptocurrency payments can be directly added to transactions in the same blockchain.

2. The presence of a trusted third party could be addressed by adding a central server to maintain the database. The SA currently employs this approach by managing a table containing the ID and global score of each user for faster access. However, fully centralizing operations at the level of such an entity would contradict the principle of transparency and limit our ability to integrate cryptocurrency payments.

3. The exposure to untrusted stakeholders and the need to restrict data modification are apparent. While blockchain participants are users registered with the CA and transactions involve goods and services declared to the PCA, it's still possible for a malicious user to attempt interference, such as reducing their own score or increasing another user's score. Therefore, modifications must be restricted, and the suitable model for this purpose is a private blockchain.

Another crucial element is the consensus protocol employed in the blockchain. For our current purpose, we will depend on a modified version of the Delegated Byzantine Fault Tolerance (dBFT) method for the following reasons:

- Keeps its coherence even when one third of participants are corrupted, further it admits to a lesser extent but still useful a certain number of corrupted delegates.
- Maintains coherence even when up to one-third of participants are corrupted, and to a lesser extent, tolerates a certain number of corrupted delegates, which remains useful.
- The system of delegation and election of representatives used helps reduce the complexity of consensus, minimize resource over-consumption, while maintaining the same level of security and consistency.

4.2.3 Communication pattern and primitives

As depicted in Figure 3.11, there are various communication patterns constituting a global model. We define a communication pattern as each pair of architectural or functional components with a set of dedicated primitives. These primitives represent atomic actions from the caller's perspective but involve a group of operations on the receiver side. Table 3.13 illustrates this model. To ensure consistency, transparency, and confidentiality preservation, no action will be permitted in the network that deviates from this communication model.

Components and actors	User	CA	PCA	SA	Blockchain participants
User		- Register	- Publish product/ service - Report a purchase		- Contribute as participant
CA	- Generate certificate - Revoke certificate		- Generate certificate - Revoke certificate	- Generate certificate - Revoke certificate	
PCA	- Generate attribute certificate - Add purchase information to attribute certificate	- Check users digital certificates - Check SA digital certificate		- Attribute certificate transmission	
SA			- Confirm transaction registration		- Push transaction to blockchain
Blockchain participants				- Confirm transaction registration	

Table 3.13: ANPA-GIEFC communication patterns

Table 3.13 provides a summary of communication patterns and should be interpreted in the direction where rows provide primitives to columns. To enhance comprehension, let's examine the specific primitives for each pair of components:

- A user can be a costumer, producer or both and can interact with:
  - Certification authority: a digital personal representative registers on behalf of a user and transmits information about the user along with a generated public key.
  - Production/ consumption authority: a user, if a producer, can publish a new product or service, and if necessary (when crypto-currency payment mode is enabled), publish the corresponding smart contract. Conversely, if the user is a customer, they should send information on the purchase operation.
  - Blockchain participants : each user participates in the blockchain; more precisely, one of their devices (the elected one) participates on their behalf.

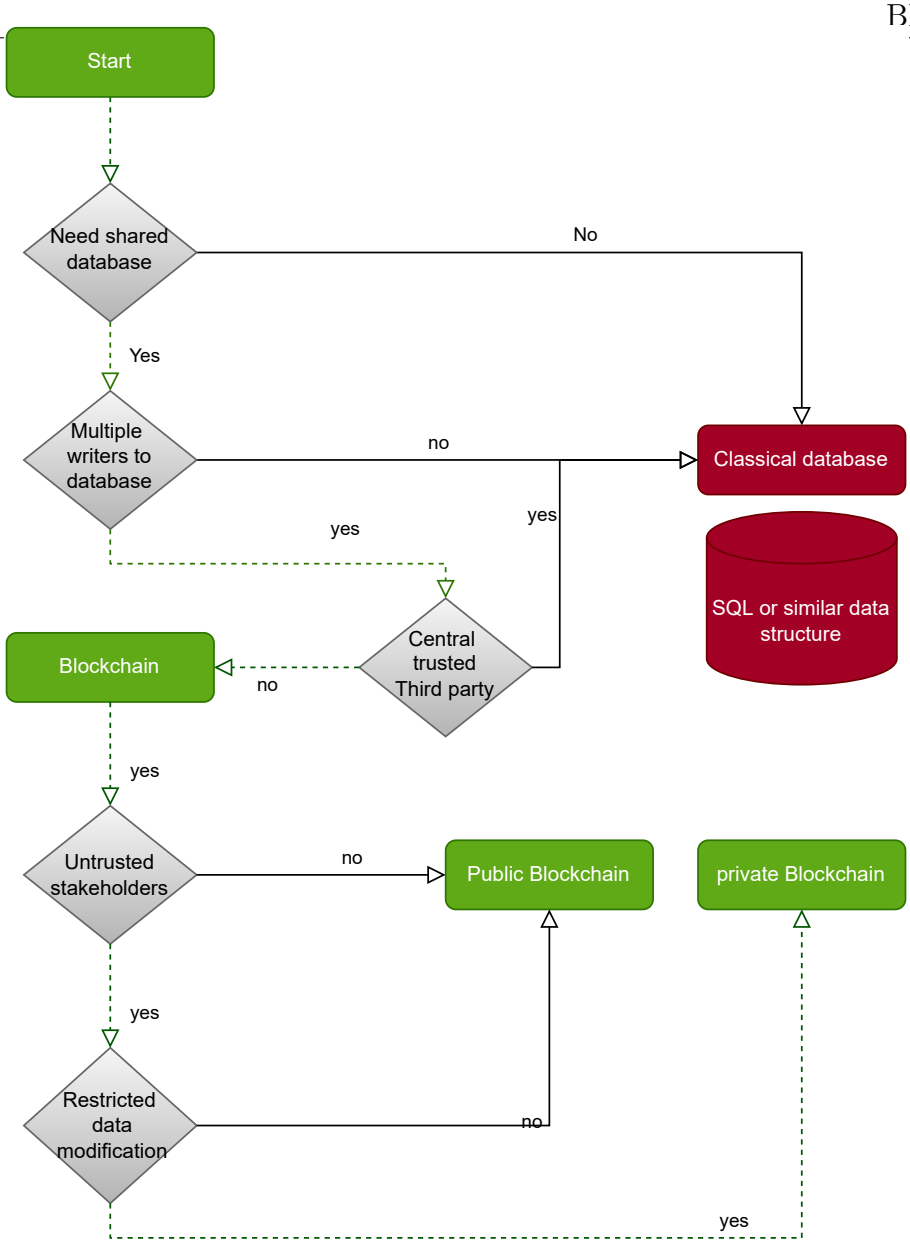

Figure 3.12: Blockchain suitability flowchart

- Certification authority provides the same pair of services to users and authorities PCA and SA. Once registered with it, it issues them a digital identity certificate. In case of an end of activities or a revelation of the key, it also handles the revocation of this certificate.
- Production/ consumption authority: as previously said PCA is in charge of production and consumption activities, to accomplish its mission in good conditions, it must incorporate the following primitives:
  - User: it generates an attribute certificate for each new good are service and then alter it by adding owner information after purchase operation.
  - CA: since it interacts with users and SA, it should be able to contact CA in order to

check identity certificate validity of each.

  - SA: After production/ acquisition operation, PCA transmit corresponding attribute certificate to scoring authority.

- Scoring authority: this instance interposes itself between the PCA and blockchain participants, it interacts with both in this way:
  - PCA: transmit information on completion of the registration of each transaction.
  - blockchain participants: pushes attribute certificate in order to register corresponding operation as a transaction.
- blockchain participants: interact exclusively with SA to verify the validity of attribute certificates. This provides the flexibility to implement a version where stakeholders can directly push attribute certificates to the blockchain and confirm the completion of transaction registration.
- Depending on whether the solution is informative, primarily used by users to assess their impacts, or legislative, employed by governmental bodies to regulate the activities of stakeholders, an additional interface with specific primitives should be established. This interface can be opened either for users or for governmental platforms, enabling the retrieval of a score per user.

4.3 Phases and algorithms

Now that we have established our assumptions, defined the functional and architectural elements involved in our proposal we can explain stages of its development.

4.3.1 User registration

Before becoming part of the ANPA-GIEFC network, individuals, whether producers or consumers, need to undergo the initial registration process. This process follows a conventional procedure, commencing with the completion of a standard form containing general user information. It concludes with the issuance of a digital identity certificate. The registration procedure is illustrated in Figure 3.13 and can be understood as follows:

1. As presumed, for the initial interaction, the user is new and has at least one physical device. To activate it, the user initiates contact with the ANPA-GIEFC platform through an auto-landing browser page that appears upon startup, similar to those used for bootstrapping Android devices.
2. The request is directed to the certification authority, which replies with its digital certificate, primarily containing the ID and public key. Additionally, the authority provides a form for the user to fill out and requests a public key proposal from the user.
3. On receiving this response:
   - The user fills into the form.
   - The device generate a pair of asymmetric keys on his behalf.

- On post command, the device builds and ciphers (using CA public key) a response message with general information and his own all new public key.

4. The registration data are transmitted to the certification authority.

5. The certification authority validates the transmitted data. In instances of governmental oversight, it is advisable to employ identifiers such as national or commercial ones to ensure the uniqueness of registrations by entity. Upon successful verification, the Certification Authority issues a digital certificate and publishes it.

6. The user receives a registration success message along with the digital certificate, signifying their admission into the network, and their device is now active.

The used certificates are X.509 like ones and contain standard information as shown in Figure 3.14, the two most important entries are:

- The ID which can be generated by using pattern of bio inspired universal unique identifier proposed by the authors in [114].
- The public key concerned by this certificate and which will be used by user for digital signature.

4.3.2 Activities reporting to PCA

After completing the registration process and having at least one active device, each user, depending on whether they are a customer, producer, or both, can suggest products, services, or engage in consumption activities. This is illustrated in Figure 3.15.

1. When a user want to publish a new service or product he must register it to the PCA:
   - He sends a request for product/ service declaration.
   - On receive the PCA first checks the validity of its certificate as legitimate service/ product provider.
   - The PCA generates an attribute certificate, primarily comprising a unique ID for each service/product, producer ID, and other descriptive information. Subsequently, the certificate is transferred to the producer.
2. The producer publishes his new service/ product on classic dedicated interfaces.
3. A costumer can then buy a published service or product.
4. Before starting to use it, he has to activate it by reporting the purchase operation to the PCA which will:
   (a) Contact CA in order to check the validity of user's certificate.
   (b) Update the attribute certificate by adding owner ID and forward it to the costumer with an active status.

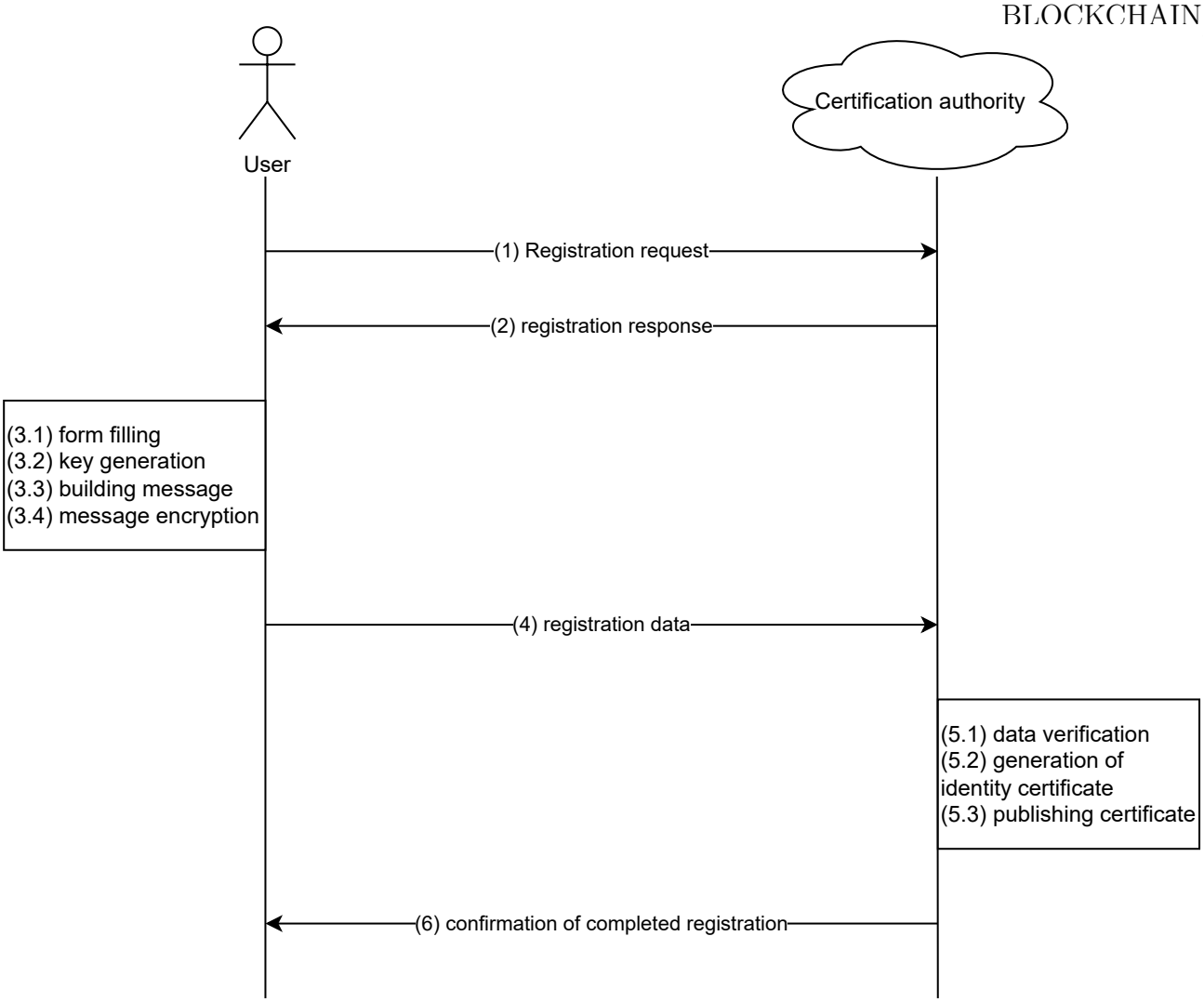

Figure 3.13: Registration stage flowchart

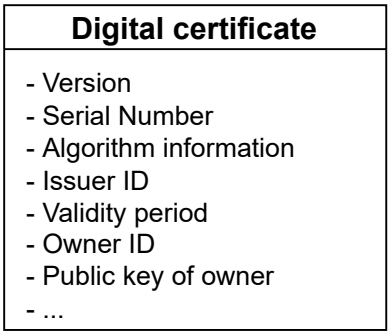

Figure 3.14: Digital certificate main structure

4.3.3 Scoring

Upon reporting an activity to the PCA, it sends an attribute certificate to the scoring authority SA, containing information regarding the publication or acquisition of a product or service.

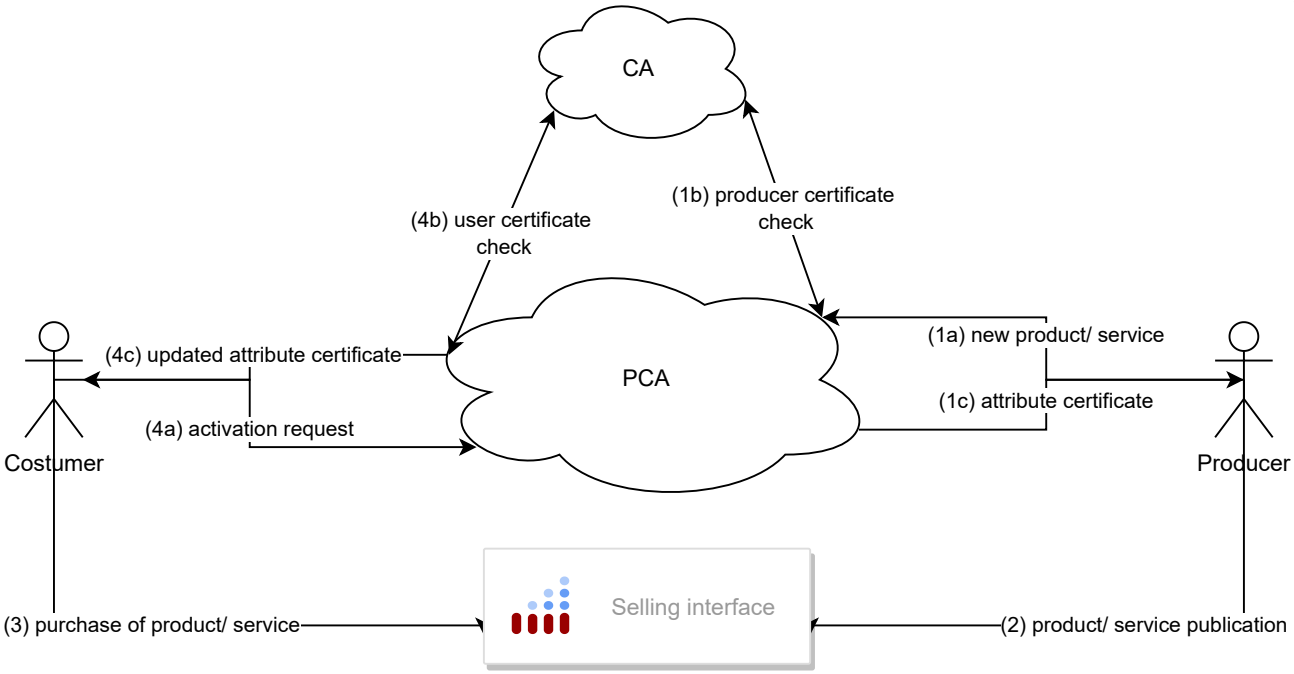

Figure 3.15: Reporting activities to PCA steps

Figure 3.16 illustrates this phase and its stages. Upon receipt, the Scoring Authority (SA) updates the score of the corresponding user. If the transmitted attribute certificate pertains to a new product or service (with a null owner ID), a production score impact is added to the existing score, dependent on the nature of the concerned good. Conversely, if the attribute certificate concerns an acquisition (with a non-null owner ID), a usage score is added to the corresponding user. Various methods exist to evaluate this score, and while it falls outside the scope of our work, it's important to note that experts propose grids considering factors such as ecological impact, measured by the quantity of CO2 emitted during the production or use of a good or service over a specified duration or distance (e.g., for cars, smartphones, air transport services, etc.).

Upon completion of these steps, the activities need to be permanently recorded in a blockchain. To function on a distributed ledger, the approach must incorporate methods that enable:

- In a prior study [6], we introduced a fully distributed and dependable leader election algorithm designed for isolated and fully automated networks. We suggest employing this same mechanism to designate representatives among all participants. Our previous research demonstrated that this approach is both lightweight and reliable compared to conventional methods. It provides the advantage of organizing nodes in a sequential list of delegates, which can then assume responsibility for validating transactions
- Define the structure of transactions and blocks: We suggest treating each attribute certificate as a transaction and capping the length of a block at a thousand operations.
- Achieve consensus (DBFT): This selection can be justified based on two factors: (i) the

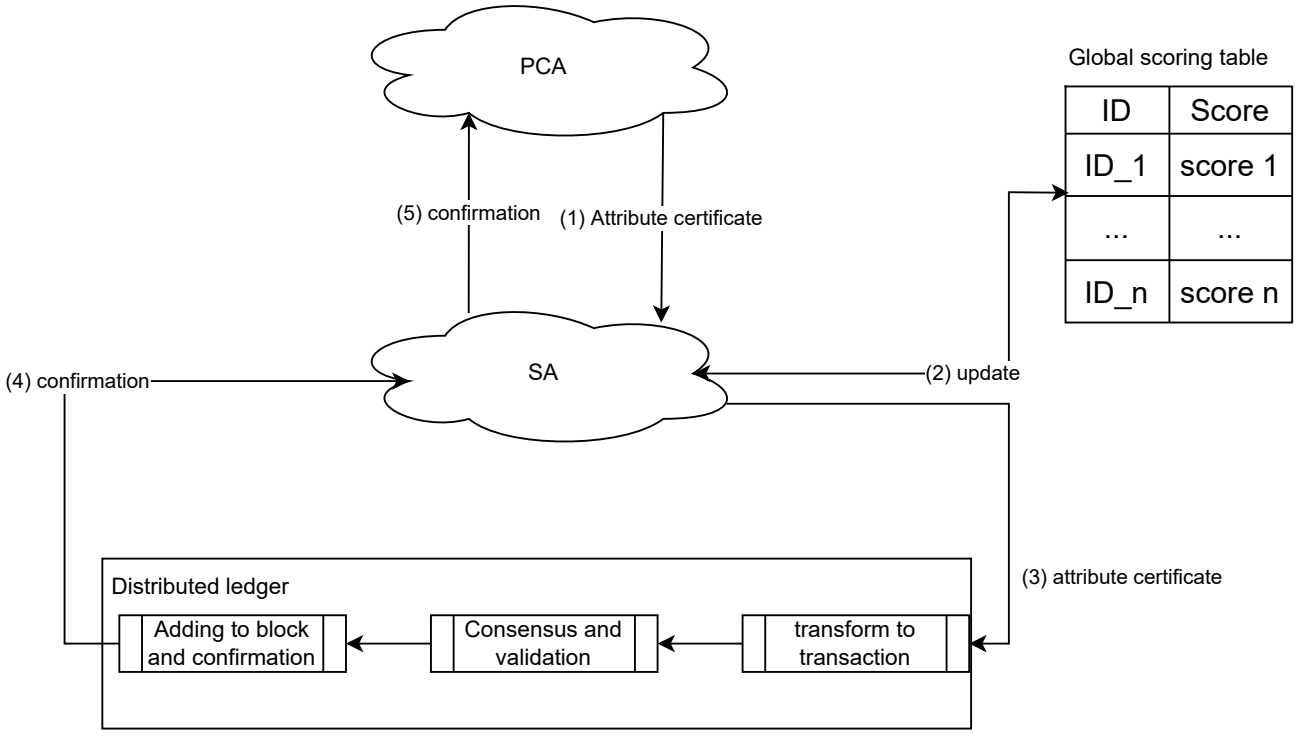

Figure 3.16: ANPA-GIEFC scoring phase

distributed nature of DBFT, which relies on representative elements, and (ii) this consensus approach tolerates up to a third of corrupted or faulty nodes in the network.

4.4 Validation methodology

Recent review papers indicate the absence of a standardized approach for validating such architectures or methods. There is a lack of comparative studies or readily available benchmark frameworks, and experimentation often depends on simulation and custom implementation [115].

To validate our architecture, we followed a three-step validation procedure: (i) Initially, we implemented our solution in a Python environment with the primary aim of extracting the execution time for each step in our functional model. (ii) Subsequently, we provided a validation proof using formal methods, specifically Petri networks—a graphical and mathematical tool employed to validate approaches like ours by offering a visual process to verify the achievability of objectives. Additionally, we utilized queue modeling to demonstrate that our architecture is in a steady state. (iii) Finally, we concluded the validation process with an evaluation of attack resilience. However, it is important to note that this is the first instance, to the best of our knowledge, of such a proposal, and as a result, conducting a formal comparative study is currently not feasible.

4.4.1 Implementation

To implement our solution, we utilized the Python programming language along with several standard libraries, including:

- Basic python: an exceptionally potent programming language, Python is highly extensible thanks to its abundance of libraries. With a vast community, it finds applications across diverse fields.
- SimPy: SimPy is a discrete-event simulation framework based on standard Python that operates on the concept of generator functions. It facilitates the creation of comprehensive scenarios for our architecture by providing a straightforward and fluid means of defining the entities and resources involved in its operation.
- Uuid: this is a Python module that generates unique identifiers following the RFC 4122 standard. Primarily, we will utilize UUID version 4, as recommended, to handle unique IDs for objects in our scenario.
- OwnCa: A lightweight library designed for handling identity certificates for hosts, servers, or clients.
- Statistics and Random: as implied by their name, these are libraries specifically designed for statistics. They assist us in gathering metrics on scenarios and generating random values, serving as the engine that empowers the generator functions at the core of SimPy logic.

4.4.2 Formal approaches

Petri networks, mathematical and graphical modeling languages, serve as a powerful tool for validating systems with simultaneous or concurrent tasks [116]. They view the systems in question as a collection of states known as places connected by transitions. A marking system allows tracking the progress and feasibility of specific activities. Validating a protocol or architecture using a Petri net involves demonstrating that, starting from a coherent initial state, one reaches a stable marking that enables the completion of all desired tasks within a specified time frame. We have represented the operation of GEFC through a Petri net, as illustrated in Figure 3.17, with positions and transitions detailed in Table 3.14.

Petri nets offer the advantage of easy readability and interpretation of the model, facilitated by three key elements: places, transitions, and the initial marking. As mentioned earlier, the graph is depicted in Figure 3.17, and the components for interpretation are provided in Table 3.14. By combining these elements, we arrive at the following step-by-step interpretation:

- The system starts with arrival of the first user at position $p0$.
- The registration phase initiates at transition $t0$ and concludes with the issuance of digital identity certificates. At the conclusion of this phase, users find themselves in either position $p2$ or $p3$, according their roles as producers or consumers, respectively.

- After a producer has completed the registration process and intends to introduce a new product or service, they need to notify the PCA through transition $t4$. To activate this transition, the producer must first be in position $p5$, representing a producer with a verified identity.

- A customer can initiate a purchase of a product or service through transition $p6$. This transition has $p5$ and $p7$ as predecessors, indicating that the product is available and the customer's ID has been verified, respectively. Upon completion, the system moves to $p8$ signifying that the product is in the instance of the declaration and activation process.

- To be activated a product must be reported to PCA through transition $t7$ then to SA by transition $t8$.

For our purposes, we employed CPN Tools, a comprehensive framework designed for modeling and evaluating colored Petri networks. Its extensive community, encompassing both researchers and developers, positions it as a favorable choice for our modeling requirements [117].

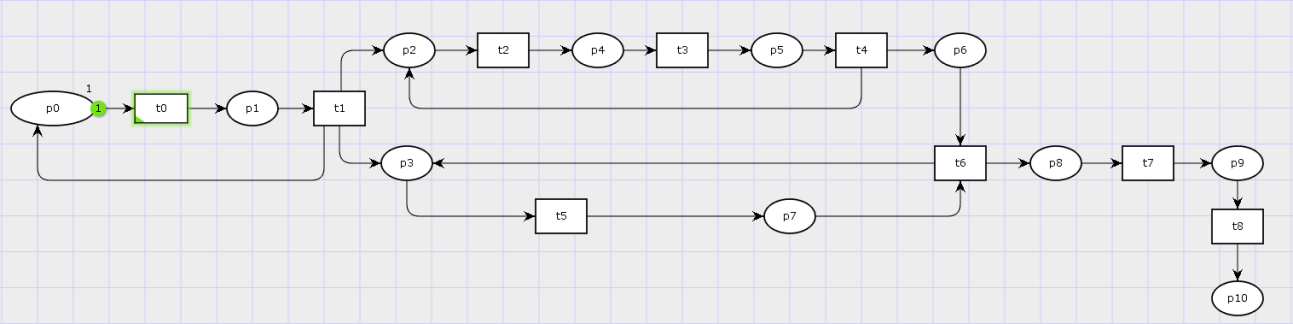

Figure 3.17: ANPA-GIEFC modeling by Petri network

Position	description	Transition	description
p0	new user arrival	t0	start registering to CA
p1	user subscribing to CA	t1	ending of registering to CA and certificate delivering
p2	active producer	t2	create product
p3	active customer	t3	check producer Id
p4	new product	t4	declare product to PCA
p5	producer Id checked	t5	check customer Id
p6	available product	t6	buy product
p7	customer Id checked	t7	register operation to PCA
p8	pre-activated product	t8	register operation to SA
p9	declared product		
p10	active product		

Table 3.14: ANPA-GIEFC Petri network positions and transitions map

Queue models are widely used mathematical tools for illustrating the stability and sustainability of systems in which components are interconnected in series or in parallel to perform complementary tasks. These components are active treatment units characterized by two parameters: queue length and service time. In a broader sense, requests arrive in a system and accumulate in a buffer, awaiting processing. Dedicated units process requests at a constant speed. The time interval between two successive arrivals to the queue is referred to as the inter-arrival rate, denoted as λ. The number of requests processed per unit of time is termed the service rate and is denoted as μ.

If we aim to examine our architecture and its functional model from a more abstract standpoint, we observe that all operations are broken down into consecutive actions that sequentially involve the authorities—the CA, followed by the acPCA, and ultimately the SA, as illustrated in Figure 3.18. The only parameter lacking a precise measurement is the arrival rate to the first queue corresponding to the CA, which we assume follows a Poisson distribution with a parameter of λ= 1/5. Additionally, we suppose that all queues are singular and of infinite size. Regarding the obtained results presented in Table **??** and taking into account that, on average, in such a system, we perform four times more verifications than the generation of certificates (an adjustable parameter) and an equal ratio of production and purchasing, we arrive at the following respective service rates for:

- Certification authority:

$$\mu 1 = 0.8 * 0.396 + 0.2 * 0.128 = 0.342 <=> 2.924 operation/second$$

- Product/ service authority:

$$\mu 2 = 0.5 * 0.347 + 0.5 * 0.342 = 0.345 <=> 2.899 operation/second$$

- Scoring authority:

$$\mu 3 = 0.299 <=> 3.344 operation/second$$

We assume that the service time at each node follows an exponential distribution with a parameter μ, as obtained in the previous calculations. To implement and measure the performance parameters of this queue model, we utilize the R language. Specifically, we employ the Simmer package, which provides numerous advantages, facilitating the acceleration of implementation and interpretation of results. The results discussed in the following subsection are obtained by averaging over a hundred different scenarios, each lasting two hours.

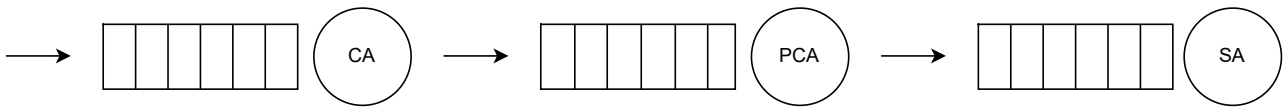

Figure 3.18: ANPA-GIEFC modeling by queue model

4.4.3 evaluation of resilience to attacks

Table 3.15 depicts the most common attacks against this kind of architecture and describes the mechanisms by which GEFC should response and stay resilient.

Attack	scenario	GEFC response mechanism
Denial-of-service	-Attacker sends a lot of requests in order to saturate servers	- A classic buffering or bashing approach should be sufficient to face this kind of attacks, a redundancy at the server level could also allow to maintain the performance when facing this kind of attack
Identity usurpation	- Attacker tries to realize social engineering attack and to steal information on users	- Identification information are centralized by the certification authority and are not accessible to any user or server, exchanges in the architecture are done by identifiers based on public cryptographic key
Data corruption	- Attacker tries to disturb scoring mechanism and to insert false transactions	- All transactions intended to be persisted and impacting user scores must go through an attribute certificate signed by the producer, the buyer and the PCA, then they have to be validated by blockchain participants
Data leakage	- Attacker tries to get information on users and their goods	- The information of the blockchain are accessible to all verified users, nevertheless a user can reconstitute the score of a profile but will not be able to link it to a customer or a company, only the legal authorities can reconstitute the complete information
Misappropriation	- Attacker tries to cash in without counterpart - Attacker takes a product or service without paying	- As the payment can be made by crypto-currency, in case of non-activation of the good or service the payment can be cancelled - On the other hand, if the payment is not made, the activation of the good can be interrupted, making it unusable and obliging the buyer to make the payment

Table 3.15: ANPA-GIEFC response mechanisms to common attacks

4.5 Results discussion

Steps/ time (seconds)	0.1	0.2	0.3	0.4	Mean time
Certificate verification	76	20	04	00	0.128
Certificate generation	00	00	02	98	0.396
Product creation	17	00	42	51	0.347
Product acquisition	00	12	34	54	0.342
Score updating	00	15	71	14	0.299

Table 3.16: ANPA-GIEFC steps timing in seconds

The primary objective of our implementation is to furnish the execution times for the major steps. These results will be employed for formal modeling. In each phase, represented in rows, we conducted a hundred iterations, categorizing for each step the number of repetitions that yielded a precise duration in seconds, as indicated in the corresponding column. These results are summarized in Table 3.16, where, for each step, we retain the average in the last column.

Marking	Transition	p0	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
0	-	1	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0
2	1	1	0	1	1	0	0	0	0	0	0	0
3	2	1	0	0	1	1	0	0	0	0	0	0
4	3	1	0	0	1	0	1	0	0	0	0	0
5	4	1	0	1	1	0	0	1	0	0	0	0
6	5	1	0	1	0	0	0	1	1	0	0	0
7	6	1	0	1	1	0	0	0	0	1	0	0
8	7	1	0	1	1	0	0	0	0	0	1	0
9	8	1	0	1	1	0	0	0	0	0	0	1

Table 3.17: Petri network marking

In order to confirm the correct operation of GIEFC, we need to prove that at any time t it is possible to:

- Subscribe to GIEFC as new user.
- Create and declare new good.
- Buy a good.

The challenge of identifying a state in the Petri net that enables the achievement of these objectives involves determining a branch in the marking graph that results in a marking equal to $(1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0)$. Table 3.17 depicts a branch in the marking graph of GIEFC that generates such a marking. It is worth noting that this marking is directly reachable from the initial mark at $p0$ by passing sequentially transitions from $t0$ to $t6$. Furthermore, once this marking is reached, it remains stable because, at the conclusion of each completed operation, a token is returned to the relevant places.

In the basic queue model of GIEFC, as depicted in Figure 3.18, it is evident that the system never reaches saturation. The cloud in Figure 3.19, where each line represents a specific scenario, illustrates that the utilization rates in all scenarios are approximately half the server capacity. Furthermore, for a more detailed insight into the evolution of server occupancy rates, the plot box in Figure 3.20 reveals that the average overall utilization rate across all scenarios and throughout the entire duration is close to 40% with minimal variation, constrained to 5%.

Figure 3.21 enables tracking the evolution of waiting time in the system, illustrating the cumulative waiting time in all queues. The blue line depicts the prevailing trend based on the average duration across all scenarios. Two key pieces of information can be gleaned: firstly, the overall waiting time in the system is less than one minute, and secondly, the waiting time remains nearly constant, represented by a curve constrained by a straight line with a y-intercept equal to one.

Another crucial and complementary graph illustrates the average duration of activities.

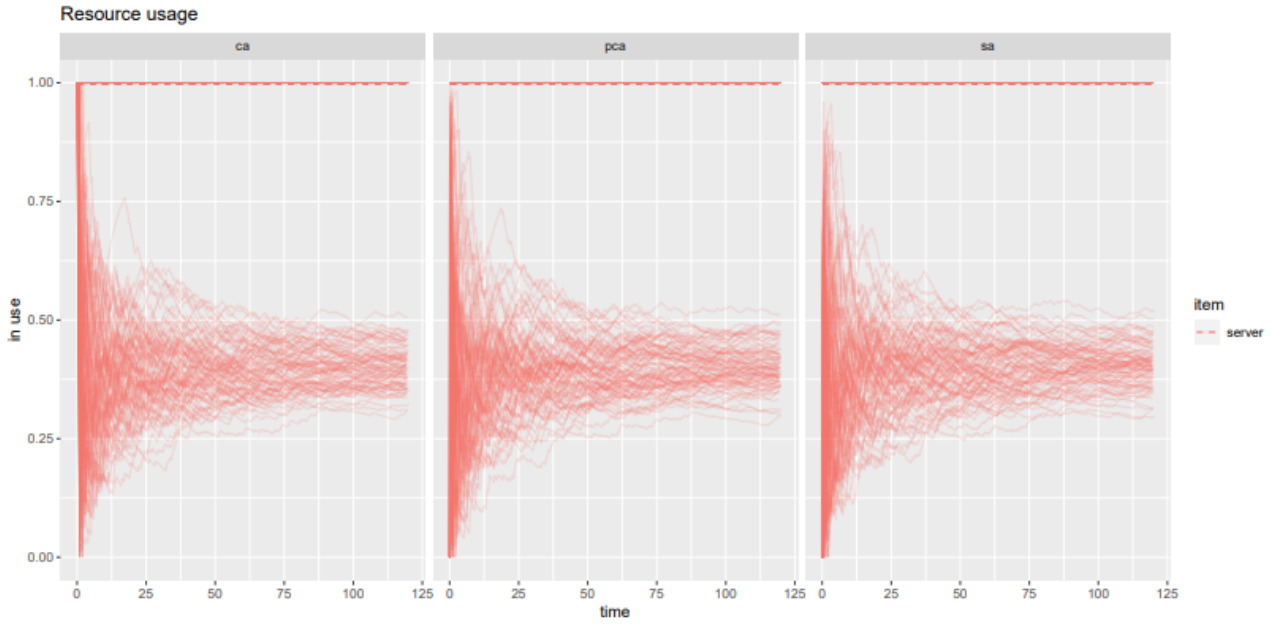

Figure 3.19: ANPA-GIEFC servers utilization rate evolution

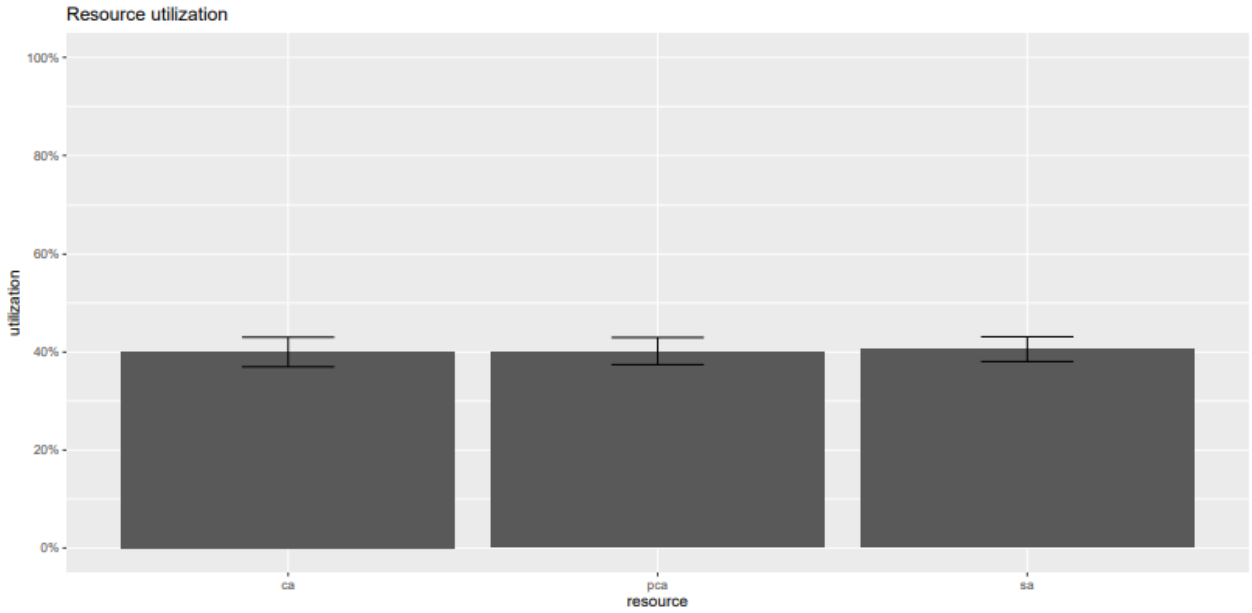

Figure 3.20: ANPA-GIEFC average servers occupation rate

Computed as the average of the sums of the execution times across the hundred scenarios considered, Figure 3.22 provides a clear view that the overall duration of activity is nearly constant. This outcome is unsurprising since it represents the sum of treatment durations, which exhibit minimal dispersion.

Flow duration, depicted by an average, reflects the overall trend of accumulated times between

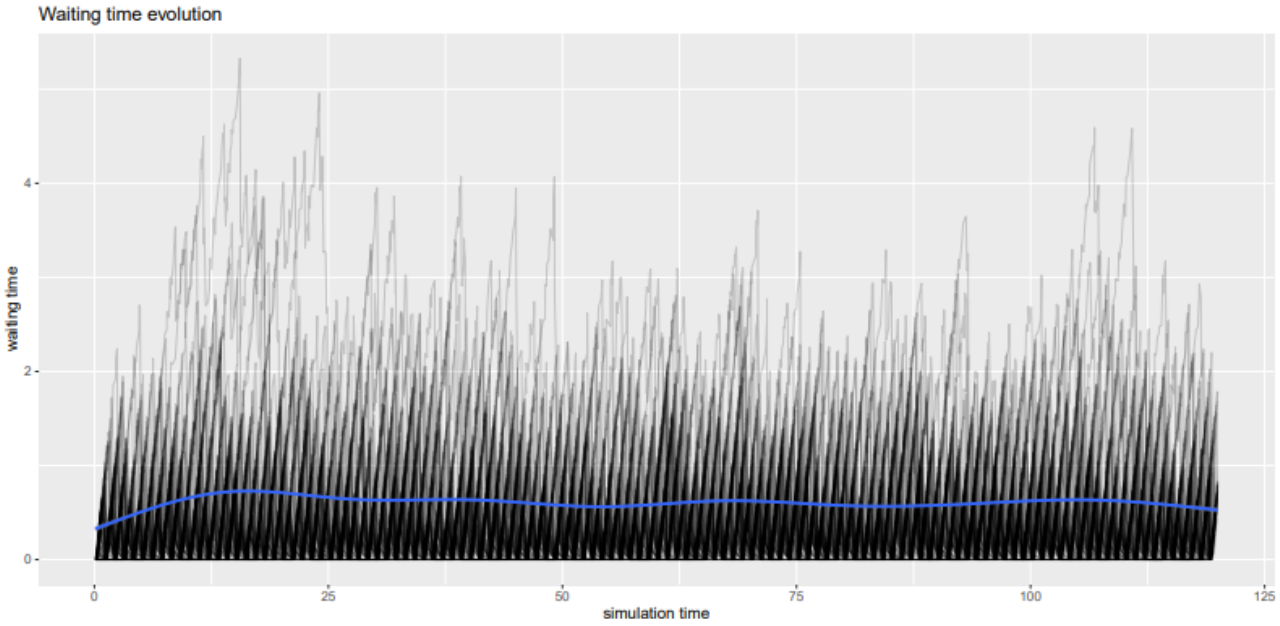

Figure 3.21: ANPA-GIEFC system waiting time evolution

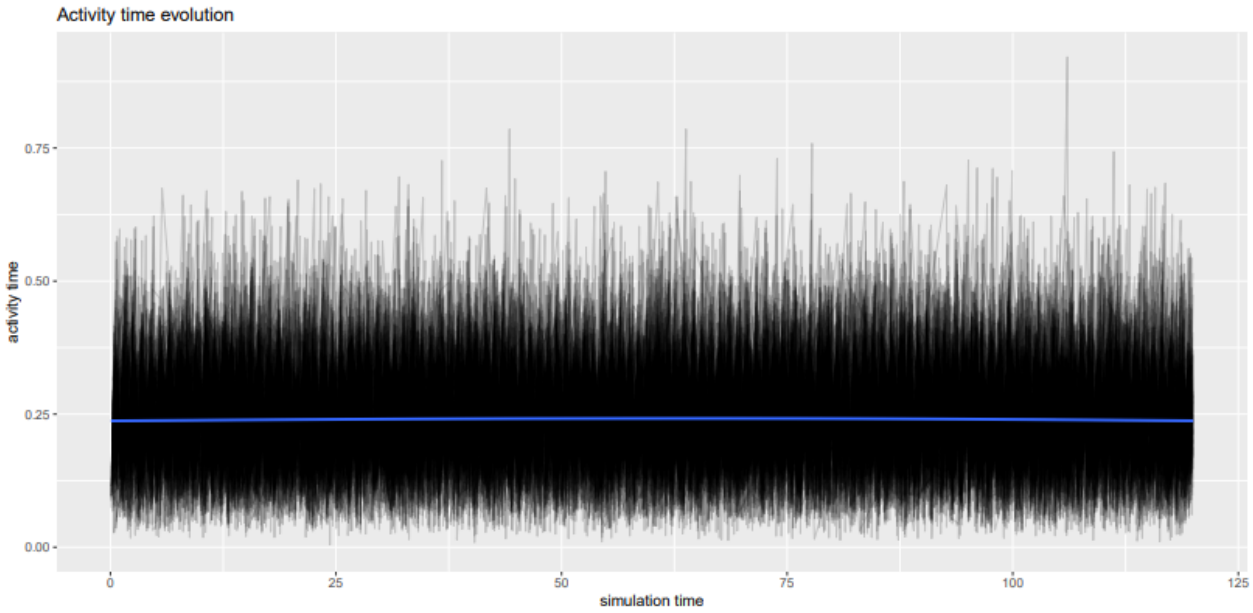

Figure 3.22: ANPA-GIEFC activity time evolution

waiting and treatment phases. As illustrated in Figure 3.23, we observe that this duration remains constant throughout the simulation time, indicating that the system remains in a steady state.

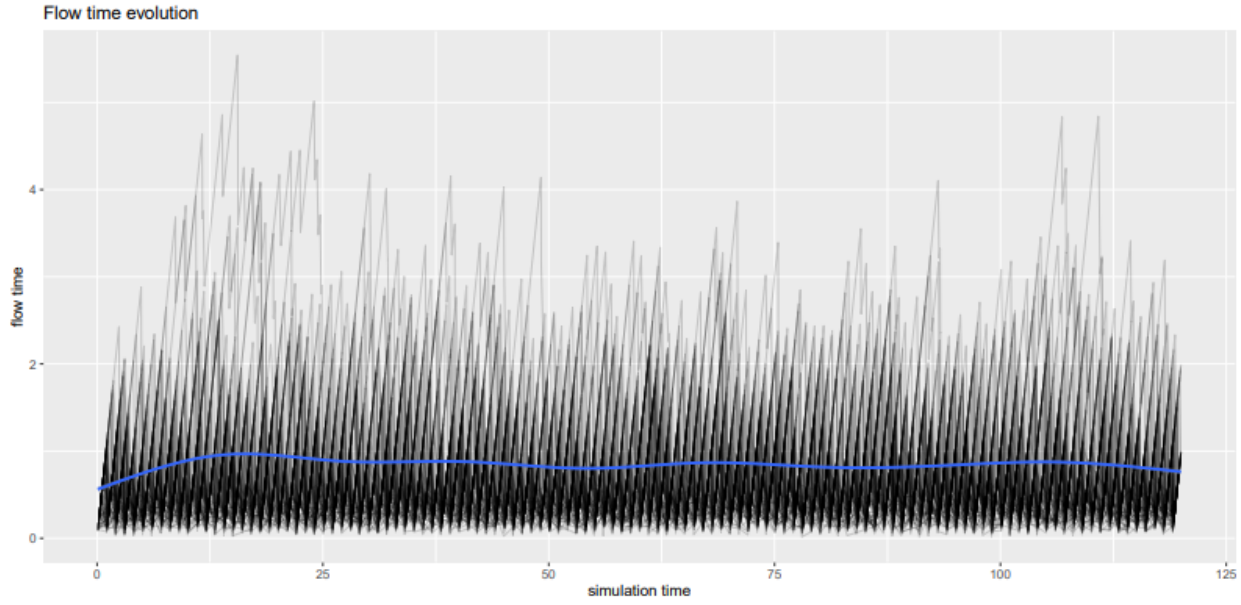

Figure 3.23: ANPA-GIEFC total flow time evolution

4.6 Conclusion

In this section we introduced a pioneering global architecture named GIEFC designed for computing ecological impact scores for both companies and customers. Leveraging widely adopted technologies such as the Internet of Things (IoT) and blockchain, our architecture encompasses key steps, including: (i) user registration with a central trusted Certification Authority CA, (ii) declaration of product or service creation/acquisition to a specialized authority known as the Production/Consumption Authority PCA, and (iii) reporting all activities along with their corresponding ecological impact scores to the Scoring Authority SA. The latter consolidates the information onto a distributed ledger maintained by users' devices.

To validate our architecture, we implemented it in a Python environment, capturing the execution times for each step. Utilizing these results, we formally represented GIEFC with a queue model to demonstrate its steady-state behavior over time evolution. Additionally, a colored Petri network was employed to illustrate that GIEFC fulfills all expected objectives outlined in the initial specifications. The performance evaluation results are highly satisfactory, providing a strong foundation for future exploration in this pioneering initiative, which, to the best of our knowledge, stands as the first of its kind.

Looking ahead to future directions, we propose a more in-depth exploration of the distributed ledger aspect. Our progress has been constrained by the limitations of existing blockchain implementation tools, which predominantly cater to cryptocurrency transactions. This restriction hindered the completion of the implementation related to recording transactions in the blockchain. Further analysis of potential attacks against this architecture is warranted, and there is a need for a more comprehensive implementation and security testing. At this juncture, we assert that the environmental issue is a governmental priority, and the landscape is conducive

to such proposals. While refinement is possible, we believe our proposal remains relevant and worthy of further development.

5 Chapter conclusion

We have discussed in this chapter the contributions that we have modestly made during our thesis work. We have contributed at two different levels on the automation problem within the cloud-centric Internet of Things and which are the embedded mechanisms for the automation of tasks on the one hand, and the self-organization of large systems deployed around of an IoT kernel.

We presented a leader election algorithm which is fully distributed and fault tolerant, it showed very competitive performance in terms of execution time and lifetime of leaders.

We have also produced a multi-level hybrid algorithm for load balancing and tasks scheduling that has shown exceptional performance and calls for reviewing how workload distribution should be done in cloud environments in the future.

Another proposal concerned the calculation of the ecological impact of individuals and societies based on no longer declarative but behavioral data . This solution was made possible thanks to two key and revolutionary technologies which are the Internet of Things and the blockchain.

This chapter concludes our thesis report, the following one will try to give an overview of what has been made during it and on the perspectives or future directions of our research work.

Conclusion and perspectives

Our thesis centered on the automation and self-organizing applications of the Internet of Things, providing us with a unique opportunity to delve into research methodology and address issues of both theoretical and practical significance. Throughout this academic endeavor, we delved into the foundational concepts of the Internet of Things (IoT) and the associated technologies that underpin it, including cloud computing and blockchain.

This exploration allowed us to gain a comprehensive understanding of the intricate interplay between these technologies and their implications for the evolving landscape of the IoT. By navigating both the theoretical underpinnings and the practical challenges, we aimed to contribute valuable insights to the broader discourse surrounding the automation and self-organizing aspects of IoT applications. In doing so, we not only enriched our own understanding but also sought to make meaningful contributions to the growing body of knowledge in this dynamic and impactful field.

At the perception layer, our investigation centered on leader election algorithms in wireless sensor networks (WSN). In the core layer, we delved into automatic load balancing and task scheduling within cloud environments. The application-level focus concluded with the development of a digital ecological impact calculator. In these areas, our modest contributions included a distributed, fault-tolerant leader election algorithm for WSN. This algorithm reduces the complexity of the leader election mechanism by distributing it across network areas, bypassing the time-consuming spanning tree building phase.

Moving to the core layer, we proposed an algorithm that hybridizes load balancing and task scheduling, optimizing their operation within the cloud environment. Our solution offers advantages such as high interoperability and decoupling of modules, reducing makespan, and minimizing SLA violations. This is achieved through operating at different levels, including regrouping servers within clusters with similar characteristics and migrating cloudlets to alleviate overloaded ones.

Finally, we introduced the first privacy-aware ecological impact calculator based on behavioral information. This tool calculates ecological impact scores for individuals and companies, serving as a crucial resource for governments in addressing climate warming issues. It enables the collection of behavioral-based ecological impact scores while respecting the privacy of companies and end consumers.

Thorough and meticulously designed experimental protocols were implemented to rigorously assess the efficacy of our proposed solutions. In direct comparison with the existing state-of-the-art works in the field, our performance demonstrated exceptional levels of satisfaction. The outcomes of these experiments not only affirmed the robustness of our approaches but also positioned them as highly competitive and impactful within the current landscape of research and innovation.

Above all, this work has given us a glimpse of the infinite possibilities open to us in terms of scientific research. Indeed, in contributing to these fundamental issues, we had to explore several paths, abandoning some and pursuing others. We believe that the problem of automation that has arisen with distributed systems is about to take on a new dimension, and that the avenues that today have not been fruitful because of the limited technological means available to us may tomorrow prove to be the scientific grail. In the future, we intend to exploit the parallelization potential of quantum computing for meta-heuristic optimization, which should enable us to better scale up our solutions. It is also obvious that we have opened the way for the exploitation of new technological paradigms to respond to the climatological crisis which is looming, more proven implementations should be able to allow us to build a proof of concept and have better chance that the solution will be adopted by targeted third parties.

Bibliography

[1] A. Rejeb, Z. Suhaiza, K. Rejeb, S. Seuring, and H. Treiblmaier, "The internet of things and the circular economy: A systematic literature review and research agenda," *Journal of Cleaner Production*, vol. 350, p. 131439, 2022.

[2] M. Soori, B. Arezoo, and R. Dastres, "Internet of things for smart factories in industry 4.0, a review," *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 192–204, 2023.

[3] Z. Alavikia and M. Shabro, "A comprehensive layered approach for implementing internet of things-enabled smart grid: A survey," *Digital Communications and Networks*, vol. 8, no. 3, pp. 388–410, 2022.

[4] F. Qaisar, H. Shahab, M. Iqbal, H. Sargana, M. Aqeel, and M. Qayyum, "Recent trends in cloud computing and iot platforms for it management and development: A review," *Pakistan Journal of Engineering and Technology*, vol. 6, pp. 98–105, Mar. 2023.

[5] A. Rejeb, S. Zailani, K. Rejeb, H. Treiblmaier, and J. G. Keogh, "Modeling enablers for blockchain adoption in the circular economy," *Sustainable Futures*, vol. 4, 2022.

[6] N. Elsakaan and K. Amroun, "Distributed and reliable leader election framework for wireless sensor network (drlef)," *International Conference on Applied CyberSecurity*, pp. 123–141, 2022.

[7] N. Elsakaan and K. Amroun, "A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment," June 2023. PREPRINT (Version 1) available at Research Square https://doi.org/10.21203/rs.3.rs-3088655/v1.

[8] N. Elsakaan and K. Amroun, "A novel privacy-aware global infrastructure for ecological footprint calculator based on the internet of things and blockchain," *The Journal of Supercomputing*, pp. 1–38, 2023.

[9] O. Ali, M. K. Ishak, M. K. L. Bhatti, I. Khan, and K. I. Kim, "A comprehensive review of internet of things: Technology stack, middlewares, and fog/edge computing interface," *Sensors*, vol. 22, 2 2022.

[10] A. E. Omolara, A. Alabdulatif, O. I. Abiodun, M. Alawida, A. Alabdulatif, W. H. Alshoura, and H. Arshad, "The internet of things security: A survey encompassing unexplored areas and new insights," *Computers and Security*, vol. 112, 1 2022.

[11] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. K. Khan, and S. K. Das, "Edge-computing-driven internet of things: A survey," *ACM Computing Surveys*, 8 2022.

[12] A. A. Laghari, K. Wu, R. A. Laghari, M. Ali, and A. A. Khan, "A review and state of art of internet of things (iot)," *Archives of Computational Methods in Engineering*, vol. 29, 2022.

[13] V. Rao and K. V. Prema, "A review on lightweight cryptography for internet-of-things based applications," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 8835–8857, 9 2021.

[14] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[15] S. Yousefi, H. Karimipour, and F. Derakhshan, "Data aggregation mechanisms on the internet of things: A systematic literature review," *Internet of Things (Netherlands)*, vol. 15, 9 2021.

[16] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the IP-based Internet of Things," *Wireless Personal Communications*, vol. 61, no. 3, pp. 527–542, 2011.

[17] H. Kopetz and W. Steiner, "Internet of things," pp. 325–341, 2022.

[18] J. Zhang, M. Ma, P. Wang, and X. dong Sun, "Middleware for the internet of things: A survey on requirements, enabling technologies, and solutions," *Journal of Systems Architecture*, vol. 117, 8 2021.

[19] T. S. Nikoui, A. M. Rahmani, A. Balador, and H. H. S. Javadi, "Internet of things architecture challenges: A systematic review," *International Journal of Communication Systems*, vol. 34, 2021.

[20] M. Lombardi, F. Pascale, and D. Santaniello, "Internet of things: A general overview between architectures, protocols and applications," *Information (Switzerland)*, vol. 12, 2021.

[21] G. Fersi, "Fog computing and internet of things in one building block: a survey and an overview of interacting technologies," *Cluster Computing*, vol. 24, 2021.

[22] D. Jain, P. V. Krishna, and V. Saritha, "A Study on Internet of Things based Applications," *arXiv.org*, vol. cs.NI, no. 1, pp. 1–10, 2012.

[23] P. Peris-lopez, J. C. Hernandez-castro, J. M. Estevez-tapiador, and A. Ribagorda, "RFID Systems : A Survey on Security Threats and Proposed Solutions," pp. 159–170, 2006.

[24] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in internet of things: Taxonomies and open challenges," *Mobile Networks and Applications*, vol. 24, 2019.

[25] H. A. Khattak, M. A. Shah, S. Khan, I. Ali, and M. Imran, "Perception layer security in internet of things," *Future Generation Computer Systems*, vol. 100, 2019.

[26] J. H. Nord, A. Koohang, and J. Paliszkiewicz, "The internet of things: Review and theoretical framework," *Expert Systems with Applications*, vol. 133, 2019.

[27] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things Characterization of Fog Computing," pp. 13–15, 2012.

[28] C. K. Emani, N. Cullot, and C. Nicolle, "ScienceDirect Understandable Big Data : A survey," *Computer Science Review*, vol. 17, pp. 70–81, 2015.

[29] O. I. Abiodun, E. O. Abiodun, M. Alawida, R. S. Alkhawaldeh, and H. Arshad, "A review on the security of the internet of things: Challenges and solutions," *Wireless Personal Communications*, vol. 119, pp. 2603–2637, 8 2021.

[30] B. Dorsemaine, J.-p. Gaulier, and P. Urien, "A new approach to investigate IoT threats based on a four layer model," 2016.

[31] M. Ahmid and O. Kazar, "A comprehensive review of the internet of things security," *Journal of Applied Security Research*, 2021.

[32] A. A. R. El-Douh, S. F. Lu, A. Elkony, and A. S. Amein, "A systematic literature review: The taxonomy of hybrid cryptography models," *Lecture Notes in Networks and Systems*, vol. 439 LNNS, pp. 714–721, 2022.

[33] I. K. Dutta, B. Ghosh, and M. Bayoumi, "Lightweight cryptography for internet of insecure things: A survey," *2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019*, pp. 475–481, 3 2019.

[34] A. Karakra and A. Alsadeh, "A-rsa: Augmented rsa," 2016.

[35] I. Chatzigiannakis, A. Pyrgelis, P. G. Spirakis, and Y. C. Stamatiou, "Elliptic curve based zero knowledge proofs and their applicability on resource constrained devices," *Proceedings - 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems, MASS 2011*, pp. 715–720, 2011.

[36] O. P. Pinol, J. E. Shahid Raza, and T. Voigt, "Bsd-based elliptic curve cryptography for theopen internet of things," 2015.

[37] K. Shahzad, T. Zia, and E. U. H. Qazi, "A review of functional encryption in iot applications," *Sensors*, vol. 22, 10 2022.

[38] M. R. Alagheband and A. Mashatan, "Advanced encryption schemes in multi-tier heterogeneous internet of things: taxonomy, capabilities, and objectives," *Journal of Supercomputing*, vol. 78, pp. 18777–18824, 11 2022.

[39] D. Kelly and M. Hammoudeh, "Optimisation of the public key encryption infrastructure for the internet of things," *ACM International Conference Proceeding Series*, 6 2018.

[40] M. Trnka, A. S. Abdelfattah, A. Shrestha, M. Coffey, and T. Cerny, "Systematic review of authentication and authorization advancements for the internet of things," *Sensors*, vol. 22, 2 2022.

[41] M. Gopala and K. Sriram, "Edge computing vs. cloud computing: An overview of big data challenges and opportunities for large enterprises," 2022.

[42] J. Hong, T. Dreibholz, J. A. Schenkel, and J. A. Hu, "An overview of multi-cloud computing," pp. 1055–1068, 2019.

[43] M. Kumar, S. C. Sharma, A. Goel, and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network and Computer Applications*, vol. 143, pp. 1–33, 10 2019.

[44] A. U. Rehman, R. L. Aguiar, and J. P. Barraca, "Fault-tolerance in the scope of cloud computing," *IEEE Access*, vol. 10, pp. 63422–63441, 2022.

[45] V. Dedeoglu, R. Jurdak, G. D. Putra, A. Dorri, and S. S. Kanhere, "A trust architecture for blockchain in iot," *ACM International Conference Proceeding Series*, 2019.

[46] B. Shrimali and H. B. Patel, "Blockchain state-of-the-art: architecture, use cases, consensus, challenges and opportunities," *Journal of King Saud University - Computer and Information Sciences*, 2021.

[47] C. H. B. Murthy, M. L. Shri, S. Kadry, and S. Lim, "Blockchain based cloud computing: Architecture and research challenges," *IEEE Access*, vol. 8, 2020.

[48] I. C. Lin and T. C. Liao, "A survey of blockchain security issues and challenges," *International Journal of Network Security*, vol. 19, 2017.

[49] J. Zheng, C. Dike, S. Pancari, Y. Wang, G. C. Giakos, W. Elmannai, and B. Wei, "An in-depth review on blockchain simulators for iot environments," *Future Internet*, vol. 14, 6 2022.

[50] X. Xu, I. Weber, M. Staples, L. Zhu, J. Bosch, L. Bass, C. Pautasso, and P. Rimba, "A taxonomy of blockchain-based systems for architecture design," *Proceedings - 2017 IEEE International Conference on Software Architecture, ICSA 2017*, 2017.

[51] S. Johar, N. Ahmad, W. Asher, H. Cruickshank, and A. Durrani, "Research and applied perspective to blockchain technology: A comprehensive survey," *Applied Sciences (Switzerland)*, vol. 11, 2021.

[52] J. Hackfeld, "A lightweight bft consensus protocol for blockchains," 3 2019.

[53] L. Alevizos, V. T. Ta, and M. H. Eiza, "Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review," *SECURITY AND PRIVACY*, vol. 5, 2022.

[54] A. Tewari and B. B. Gupta, "Security, privacy and trust of different layers in internet-of-things (iots) framework," *Future Generation Computer Systems*, vol. 108, 2020.

[55] M. binti Mohamad Noor and W. H. Hassan, "Current research on internet of things (iot) security: A survey," *Computer Networks*, vol. 148, 2019.

[56] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, "A survey of internet of things (iot) authentication schemes," *Sensors (Switzerland)*, vol. 19, 2019.

[57] A. Bounceur, M. Bezoui, R. Euler, N. Kadjouh, and F. Lalem, "Brogo: A new low energy consumption algorithm for leader election in wsns," *Proceedings - International Conference on Developments in eSystems Engineering, DeSE*, 2018.

[58] S. Vasudevan, J. Kurose, and D. Towsley, "Design and analysis of a leader election algorithm for mobile ad hoc networks," *Proceedings - International Conference on Network Protocols, ICNP*, 2004.

[59] P. Chaparala, A. R. Atmakuri, and S. S. S. Rao, "3 -phase leader election algorithm for distributed systems," *Proceedings of the 3rd International Conference on Computing Methodologies and Communication, ICCMC 2019*, 2019.

[60] M. Numan, F. Subhan, W. Z. Khan, B. Assiri, and N. Armi, "Well-organized bully leader election algorithm for distributed system," *Proceedings - 2018 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications, ICRAMET 2018*, 2018.

[61] A. Bounceur, M. Bezoui, M. Lounis, R. Euler, and C. Teodorov, "A new dominating tree routing algorithm for efficient leader election in iot networks," *CCNC 2018 - 2018 15th IEEE Annual Consumer Communications and Networking Conference*, vol. 2018-January, 2018.

[62] M. Mendez, F. G. Tinetti, A. M. Duran, D. A. Obon, and N. G. Bartolome, "Distributed algorithms on iot devices: Bully leader election," *Proceedings - 2017 International Conference on Computational Science and Computational Intelligence, CSCI 2017*, 2018.

[63] C. Sommer, F. Hagenauer, and F. Dressler, "A networking perspective on self-organizing intersection management," *2014 IEEE World Forum on Internet of Things, WF-IoT 2014*, 2014.

[64] P. Choudhary, R. K. Dwivedi, and U. Singh, "Novel algorithm for leader election process in virtual traffic light protocol," *International Journal of Information Technology (Singapore)*, vol. 12, 2020.

[65] R. Elchamaa, M. Gueriau, B. Dafflon, R. K. Chamoun, and Y. Ouzrout, "A local leader election protocol applied to decentralized traffic regulation," *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, vol. 2017-November, 2018.

[66] P. Pace, G. Aloi, G. Caliciuri, and G. Fortino, "Management and coordination framework for aerial-terrestrial smart drone networks," *SmartObjects 2015 - Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects, co-located with MobiCom 2015*, 2015.

[67] Y. Sudo, F. Ooshita, T. Izumi, H. Kakugawa, and T. Masuzawa, "Time-optimal leader election in population protocols," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, 2020.

[68] A. Bounceur, M. Bezoui, R. Euler, F. Lalem, and M. Lounis, "A revised brogo algorithm for leader election in wireless sensor and iot networks," pp. 1–3, 2017.

[69] M. K. Murmu and A. K. Singh, "A bio-inspired leader election protocol for cognitive radio networks," *Cluster Computing*, vol. 22, 2019.

[70] H. Alazzam, W. Mardini, A. Alsmady, and A. Enizat, "Load balancing in cloud computing using water flow-like algorithm," *ACM International Conference Proceeding Series*, 2019.

[71] T. M. Tawfeeg, A. Yousif, A. Hassan, S. M. Alqhtani, R. Hamza, M. B. Bashir, and A. Ali, "Cloud dynamic load balancing and reactive fault tolerance techniques: A systematic literature review (slr)," *IEEE Access*, vol. 10, pp. 71853–71873, 2022.

[72] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Computing Surveys*, vol. 51, 2019.

[73] S. Souravlas, S. D. Anastasiadou, N. Tantalaki, and S. Katsavounis, "A fair, dynamic load balanced task distribution strategy for heterogeneous cloud platforms based on markov process modeling," *IEEE Access*, vol. 10, pp. 26149–26162, 2022.

[74] T. Deepa and D. D. Cheelu, "A comparative study of static and dynamic load balancing algorithms in cloud computing," *Proceedings of International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS-2017)*, 2017.

[75] T. Kokilavani and D. I. G. Amalarethinam, "Load balanced min-min algorithm for static meta-task scheduling in grid computing," *International Journal of Computer Applications*, vol. 20, pp. 975–8887, 2011.

[76] G. Liu, J. Li, and J. Xu, "Aisc 191 - an improved min-min algorithm in cloud computing," *AISC*, vol. 191, pp. 47–52, 2012.

[77] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, and X. Xu, "A discrete pso-based static load balancing algorithm for distributed simulations in a cloud environment," *Future Generation Computer Systems*, vol. 115, pp. 497–516, 2 2021.

[78] V. Arulkumar and N. Bhalaji, "Performance analysis of nature inspired load balancing algorithm in cloud environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 3735–3742, 3 2021.

[79] S. T. Milan, L. Rajabion, H. Ranjbar, and N. J. Navimipour, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments," *Computers and Operations Research*, vol. 110, pp. 159–187, 2019.

[80] K. R. R. Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud," *Advances in Intelligent Systems and Computing*, vol. 424, pp. 67–78, 2016.

[81] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G. B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *Journal of Supercomputing*, vol. 76, pp. 6302–6329, 8 2020.

[82] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in iaas cloud," *Journal of Network and Computer Applications*, vol. 128, pp. 64–77, 2 2019.

[83] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021.

[84] M. Vanitha and P. Marikkannu, "Effective resource utilization in cloud environment through a dynamic well-organized load balancing algorithm for virtual machines," *Computers and Electrical Engineering*, vol. 57, pp. 199–208, 1 2017.

[85] M. Kumar and S. C. Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment," *Computers and Electrical Engineering*, vol. 69, pp. 395–411, 7 2018.

[86] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Applied Soft Computing Journal*, vol. 76, pp. 416–424, 3 2019.

[87] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep q-learning," *Information Sciences*, vol. 512, pp. 1170–1191, 2 2020.

[88] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 2332–2342, 6 2022.

[89] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," *IEEE Access*, vol. 7, pp. 42735–42744, 2019.

[90] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "Faco: a hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 3975–3987, 10 2020.

[91] D. Chaudhary and B. Kumar, "Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing," *Applied Soft Computing Journal*, vol. 83, 10 2019.

[92] A. Satpathy, S. K. Addya, A. K. Turuk, B. Majhi, and G. Sahoo, "Crow search based virtual machine placement strategy in cloud data centers with live migration," *Computers and Electrical Engineering*, vol. 69, pp. 334–350, 7 2018.

[93] A. S. Abohamama, M. F. Alrahmawy, and M. A. Elsoud, "Improving the dependability of cloud environment for hosting real time applications," *Ain Shams Engineering Journal*, vol. 9, pp. 3335–3346, 12 2018.

[94] H. Han, W. Bao, X. Zhu, X. Feng, and W. Zhou, "Fault-tolerant scheduling for hybrid real-time tasks based on cpb model in cloud," *IEEE Access*, vol. 6, pp. 18616–18629, 2 2018.

[95] H. Sun, H. Yu, G. Fan, and L. Chen, "Qos-aware task placement with fault-tolerance in the edge-cloud," *IEEE Access*, vol. 8, pp. 77987–78003, 2020.

[96] S. Chinnathambi, A. Santhanam, J. Rajarathinam, and M. Senthilkumar, "Scheduling and checkpointing optimization algorithm for byzantine fault tolerance in cloud clusters," *Cluster Computing*, vol. 22, pp. 14637–14650, 11 2019.

[97] X. Xu, R. Mo, F. Dai, W. Lin, S. Wan, and W. Dou, "Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud," *IEEE Transactions on Industrial Informatics*, vol. 16, pp. 6172–6181, 9 2020.

[98] P. C. Michael Schukat, "Public key infrastructures and digital certificates for the Internet of things," *2015 26th Irish Signals and Systems Conference, ISSC 2015*, no. ii, 2015.

[99] P. R. Shukla, J. Skea, A. Reisinger, R. Slade, R. Fradera, M. Pathak, A. Al, K. Malek, B. R. V. Diemen, A. Hasija, G. Lisboa, S. Luz, J. Malley, D. Mccollum, and S. Some, "Mitigation of climate change : Working group iii contribution to the sixth assessment report of the intergovernmental panel on climate change," 2022.

[100] Unfccc, "Report of the conference of the parties on its twenty-first session, held in paris from 30 november to 11 december 2015. part one: Proceedings."

[101] "Gender, climate change and health," 2014.

[102] "Governing rapid climate mitigation mark diesendorf unsw sydney,"

[103] J. Mulrow, K. Machaj, J. Deanes, and S. Derrible, "The state of carbon footprint calculators: An evaluation of calculator design and user interaction features," *Sustainable Production and Consumption*, vol. 18, pp. 33–40, 4 2019.

[104] J. Yan, Q. Lu, J. Tang, L. Chen, J. Hong, and T. Broyd, "Digital tools for revealing and reducing carbon footprint in infrastructure, building, and city scopes," *Buildings*, vol. 12, 8 2022.

[105] N. Elsakaan and K. Amroun, "A comparative study of machine learning binary classification methods for botnet detection," in *International Conference on Applied CyberSecurity*, pp. 20–34, Springer, 2021.

[106] P. Y. Chen and A. O. Hero, "Local fiedler vector centrality for detection of deep and overlapping communities in networks," 2014.

[107] A. Casteigts, "Jbotsim: A tool for fast prototyping of distributed algorithms in dynamic networks," *SIMUTOOLS 2015 - 8th EAI International Conference on Simulation Tools and Techniques*, 2015.

[108] T. Kavitha, S. Hemalatha, T. Saravanan, A. K. Singh, M. I. Alam, and S. Warshi, "Survey on cloud computing security and scheduling," pp. 1–4, 2022.

[109] N. Elsakaan and K. Amroun, "A novel multi-level hybrid load balancing and tasks scheduling algorithm for cloud computing environment," *The Journal of Supercomputing*, pp. 1–41, 2024.

[110] T. Goyal, A. Singh, and A. Agrawal, "Cloudsim: simulator for cloud computing infrastructure and modeling," *Procedia Engineering*, vol. 38, pp. 3566–3572, 2012. INTERNATIONAL CONFERENCE ON MODELLING OPTIMIZATION AND COMPUTING.

[111] M. C. Filho, R. L. Oliveira, C. C. Monteiro, P. R. Inácio, and M. M. Freire, "Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," *Proceedings of the IM 2017 - 2017 IFIP/IEEE International Symposium on Integrated Network and Service Management*, 2017.

[112] M. A. Vasile, F. Pop, R. I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Generation Computer Systems*, vol. 51, pp. 61–71, 10 2015.

[113] V. Hassija, S. Zeadally, I. Jain, A. Tahiliani, V. Chamola, and S. Gupta, "Framework for determining the suitability of blockchain: Criteria and issues to consider," *Transactions on Emerging Telecommunications Technologies*, vol. 32, 2021.

[114] D. Mariyanayagam, P. Shukla, and B. S. Virdee, "Bio-inspired framework for security in iot devices," pp. 749–757, 2022.

[115] C. Wang, X. Cheng, J. Li, Y. He, and K. Xiao, "A survey: applications of blockchain in the internet of vehicles," *Eurasip Journal on Wireless Communications and Networking*, vol. 2021, 2021.

[116] W. J. Thong and M. A. Ameedeen, "A survey of petri net tools," *Lecture Notes in Electrical Engineering*, vol. 315, pp. 537–551, 2015.

[117] V. Faerman, K. Voevodin, and V. Avramchuk, "Case of discrete-event simulation of the simple sensor node with cpn tools," pp. 1–9, 1 2023.

Abstract

The last decade has seen the rise of a set of information technologies, ranging from the expansion of the traditional Internet to interconnect smart objects, big data, cloud computing to the blockchain. As these technologies have evolved, they have converged to rise a new paradigm known as the Internet of Things. In its cloud-centric architecture, the Internet of Things links billions of objects and users worldwide. This has motivated researchers to focus on the problem of automating actions such as connecting new objects to the network, or self-organizing underground components such as wireless sensor networks. Our thesis focus on these automation mechanisms which aim to reduce human intervention to ensure functional safety and availability over time. At various levels we have identified crucial modules, such as the leader election at the perception layer, task scheduling and load balancing in the cloud environment at the core layer, or even more complex at the application layer, the large-scale deployment of a self-managed platform for calculating the ecological footprints of companies and individuals. We carried out a state-of-the-art survey, a comparative and assessment study of the most recent solutions, before proposing our own approaches. Experimentation has shown very satisfactory results, making our works to be among the most relevant ones in the current literature.

Keywords: Internet of things, cloud computing, blockchain, wireless sensor network, automation mechanisms, load balancing, tasks scheduling, clustering, leader election, ecological footprint, genetic algorithm.

Résumé

La dernière décennie a connu l'essor d'un ensemble révolutionnaire de technologies de l'information, parmi elles on peut retrouver l'élargissement de l'internet classique pour interconnecter des objets intelligents, le big data, le cloud computing ou encore la blockchain. Au fil de leurs évolutions ses technologies ont convergé et ont permis la naissance d'un nouveau paradigme appelé l'internet des objets. Dans son organisation centrée sur le cloud, l'internet des objets interconnecte des milliards d'objets et d'utilisateurs à travers le monde. Ceci a incité les chercheurs à s'intéresser à la problématique de l'automatisation des actions comme l'intégration au réseau de nouveaux objets ou encore comme l'auto-organisation des parties dites ensevelies comme les réseaux de capteurs sans fil. Notre thèse s'intéresse à ces mécanismes d'automatisation visant à réduire l'intervention humaine dans l'assurance de la sureté fonctionnelle et de la continuité de délivrance des services. À différents niveaux nous avons identifié des modules cruciaux, comme le leader élection au niveau de la couche perception, de la planification des tâches et de l'équilibrage des charges dans le cloud au niveau de la couche core, ou encore plus complexe au niveau des applications, le déploiement à large échelle d'une plateforme auto-gérée pour le calcul des impacts écologiques des entreprises et des individus. Nous avons réalisé une étude de l'état de l'art, une étude comparative et critique des solutions les plus récentes avant de proposer nos propres approches. Des expérimentations ont permis d'obtenir des résultats très satisfaisants positionnant de ce fait nos travaux comme étant parmi les plus pertinents dans la littérature actuelle.

Mots-clés: Internet des objets, cloud computing, blockchain, réseaux de capteurs sans fil, mécanismes d'automatisation, équilibrage de charge, planification des tâches, clustering, élection de leader, empreinte écologique, algorithme génétique.

خلاصة

شهد العقد الماضي ظهور مجموعة من تقنيات المعلومات ، بدءًا من التوسع في الإنترنت التقليدي إلى ربط الأشياء الذكية والبيانات الضخمة والحوسبة السحابية وصولا إلى البلوك تشين. مع تطور هذه التقنيات ، تقاربت لتظهر نموذجًا جديدًا يُعرف باسم إنترنت الأشياء. في بنيته التي تركز على السحابة يربط إنترنت الأشياء مليارات الأشياء والمستخدمين في جميع أنحاء العالم. وقد حفز هذا الباحثين على التركيز على مشكلة أتمتة الإجراءات مثل توصيل أشيا جديدة بالشبكة ، أو ذاتية التنظيم لدى مكونات تحت الأرض مثل شبكات الاستشعار اللاسلكية. تركز أطروحتنا على آليات الأتمتة هذه التي تهدف إلى تقليل التدخل البشري لضمان السلامة الوظيفية والتوافر بمرور الوقت. على مستويات مختلفة ، حددنا الوحدات النمطية الحاسمة ، مثل انتخاب القائد في طبقة الإدراك ، وجدولة المهام وموازنة الحمل في بيئة السحابة في الطبقة الأساسية ، أو حتى أكثر تعقيدًا في طبقة التطبيق ، النشر على نطاق واسع منصة مُدارة لحساب البصمات البيئية للشركات والأفراد. لقد أجرينا مسحًا على أحدث طراز ، ودراسة مقارنة وتقييمية لأحدث الحلول ، قبل اقتراح مناهجنا الخاصة. أظهرت التجارب نتائج مرضية للغاية ، مما يجعل أعمالنا من بين الأعمال الأكثر صلة بالموضوع في الأدبيات الحالية.

الكلمات المفتاحية: إنترنت الأشياء ، الحوسبة السحابية ، البلوك تشين ، شبكة الاستشعار اللاسلكية ، آليات التشغيل الآلي ، موازنة الحمل جدولة المهام ، التجميع ، انتخاب القائد ، البصمة البيئية ، الخوارزمية الجينية.