REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Ferhat Abbas Sétif
Faculté des Sciences
Département d'Informatique

Les Réseaux de Capteurs sans Fil dans l'Internet des Objects

Thèse présentée par :

Sarra HAMMOUDI

En vue de l'obtention du diplôme de

Doctorat $3^{ème}$ cycle LMD en Informatique

Soutenue devant le jury composé de :

Nadjet	KAMEL	Prof. Université Ferhat Abbas Sétif 1	Président
Zibouda	ALIOUAT	MCA. Université Ferhat Abbas Sétif 1	Directrice
Saad	HAROUS	Prof. Université United Arab Emirates	Co-directeur
Faouzi	SEMCHEDDINE	Prof. Université Ferhat Abbas Sétif 1	Examinateur
Omar	MAWLOUD	MCA.Université A. Mira Béjaia	Examinateur
Mourad	GUEROUI	MCA. Université Versailles Saint-Quentin-en-Yvelines Paris	Invité

2019

PEOPLE'S DEMOCRATIC REPUBLIC of ALGERIA
Ministry of Higher Education and Scientific Research
Ferhat Abbas University of Sétif
Faculty of Sciences
Department of Computer Science

Wireless Sensor Networks for Internet of Things

by

Sarra HAMMOUDI

A thesis submitted in fulfillment of the requirement for the degree of PhD

Board of examiners :

Nadjet	KAMEL	Prof. Ferhat Abbas Setif 1 University	President
Zibouda	ALIOUAT	MCA. Ferhat Abbas Setif 1 University	Supervisor
Saad	HAROUS	Prof.United Arab Emirates University	Co-supervisor
Faouzi	SEMCHEDDINE	Prof. Ferhat Abbas Setif 1 University	Examinator
Omar	MAWLOUD	MCA. A. Mira Bejaia University	Examinator
Mourad	GUEROUI	MCA. Versailles Saint-Quentin-en-Yvelines, Paris University	Invited

2019

Declaration

I, Sarra HAMMOUDI, declare that this thesis, " Wireless Sensor Networks over Internet of Things ", and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that :

1. This work was done wholly or mainly while in candidature for a research degree at this University ;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated ;
3. Where I have consulted the published work of others, this is always clearly attributed ;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work ;
5. I have acknowledged all main sources of help ;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself ;

Signed : .....

Date : .....

Acknowledgments

In the name of ALLAH, the most Gracious and the most Merciful. I thank ALLAH for the Strength and Blessing He has Given me to achieve this work.

First of all I express my heartfelt thanks to my supervisor **Dr. Zibouda ALIOUAT** and my co-supervisor **Prof. Saad HAROUS**, for their insightful instructions during the development of this thesis. Their guidance and comments on the text were a tremendous academic support for me.

My most sincere gratitude goes to **Dr. Lemia Louail** for her helpful suggestions.

I also appreciate the availability and the guidance of **Dr. Guerroui Mourad**, especially during my presence at LI-PaRAD laboratory, University Versailles.

I extend my sincere thanks to all members of the Department of Computer science, and all my teachers who contributed directly or indirectly to the successful studies.

My parents have been an inspiration throughout my life. They always support and encourage me to reach my dreams and aspirations. May ALLAH reward them.

I wish to thank the members of my dissertation committee who were more than generous with their expertise, guidance and precious time throughout the review of this document.

I thank the College of Information Technology of UAE university and LI-PaRAD laboratory of University of Versailles for hosting during my internships.

Abstract

With the emerging technology of the Internet of Things (IoT), a huge number of intelligent devices are being connected to the Internet. Wireless Sensor Networks and Cloud computing are the main elements that facilitate the emergence of IoT. Sensors storage capacity and power consumption are among the main challenging issues in IoT. The 2.4 GHZ band authorizes many different wireless technologies to exploit the same spectrum, which yields a severe interference environment. The IEEE 802.15.4 proposed the Time Slotted Channel Hopping (TSCH) designed for low-power and lossy networks (LLNs). The TSCH aims to improve the reliability of sensed data transmission by adapting the channel hopping technique to alleviate the external interference. Because channels can be affected by Wi-Fi signals, hopping blindly from one channel to another may also deteriorate the data transmission performance. To solve this problem, we propose a new strategy dedicated to TSCH that makes sensors intelligent and enables them to hop channels in a wise manner. TSCH uses shared links to increase the networks' throughput. To ensure collision avoidance in the presence of hidden nodes, this thesis proposes two intelligent algorithms : Time Slotted Channel Hopping with Correct Collision Avoidance backoff algorithm(TSCH-CCA) and Enhanced Priority Channel Access Backoff Algorithm(E-PCA), applied respectively to both normal packets and critical events packets. Our proposed solution shows significant improvements in terms of latency, network congestion, network lifetime, critical event packets lifetime, and collision avoidance. We also proposed two algorithms that overcome the external interference and multi-path fading. These algorithms are called : Enhanced Time Slotted Channel Hopping (E-TSCH) and Reliable Time Slotted Channel Hopping (R-TSCH). To evaluate their performance, we implemented them in Network Simulator 3 (NS3) and compared them to TSCH. The results show significant improvements in terms of the retransmitted packets number, packet delivery ratio, and energy consumption. We also proposed three IaaSs on Cloud servers that seek to

ensure Load Balancing, minimize the clients latency and provide a fault tolerant system by ensuring critical data availability and achieving fast input and output critical requests.

keywords : IoT, Big data, Infrastructure as a Service, Load Balancing, Cloud computing, multi-Agent systems, IEEE 802.15.4e, TSCH, MAC sub-layer, Smart sensors, WSNs.

Table of contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

IoT	Internet of Things
TSCH	Time Slotted Channel Hopping
LLNs	low-power and lossy networks
TSCH-CCA	Time Slotted Channel Hopping with Correct Collision Avoidance
E-PCA	Enhanced Priority Channel Access Backoff Algorithm
NS3	Network Simulator 3
WSN	Wireless Sensor Networks
RFID	Radio Frequency IDentification
TTP	Time Triggered Protocol
ETP	Event Triggered Protocol
LBCS	Load Balancing in the Cloud Using Specialization
ISM	Industrial Scientific and Medical
CO	carbon monoxide
CO2	carbon dioxide
ASIC	Application specific integrated circuit
FPGA	special-purpose field programmable gate array
MCUs	microcontrollers
ITU	The International Telecommunication Union
IBSG	Internet Business Solutions Group
UUID	Universal Unique IDentifier
QoS	Quality of Service
APIs	application programming interfaces
IFP	Interface Profile
UpnP	Universal Plug and Play
SOA	Service Oriented Architecture

UHF	Ultra High Frequency
URN	Uniform Resource Name
CoAP	Constrained Application Protocol
CoRE	Constrained RESTful Environments
REST	REpresentational State Transfer
HTTP	Hypertext Transfer Protocol
URI	Uniform Resource Identifier
UDP	User Datagram Protocol
6LoWPAN	IPv6 over Low power WPAN
MTU	Pv6 Maximum Transmission Unit
RFD	Reduced Function Devices
DSME	Deterministic and Synchronous Multi-channel Extension
AMCA	Asynchronous multichannel adaptation
LLDN	Low Latency Deterministic Network
BLINK	Radio Frequency Identification Blink
DSN	Data Stream Network
OGC	Open Geospatial Consortium
SWE	Sensor Web Enablement
SOS	Sensor Observation Service
NIST	The National Institute of Standard and Technologies
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
IoT-A	Internet of Things-Architecture
PUF	Physical Unclonable Functions
HPCs	Hardware Performance Counters
SCC	smart and connected communities
TDMA	Time Division Multiple Access
CSMA	Carrier Sense Multiple Access
FDMA	Frequency Division Multiple Access
CWS	compressive wireless sensing
BDSMC	Big data stream mobile computing
B-MINet	broadband mobile Internet networking

Rt-MCC	real-time mobile Cloud computing
VM	Virtual Machine
CQRM	Coding-aware QoS Routing Metric
M2M	Machine to Machine
LWM2M	Lightweight M2M
NETCONF	Network Configuration Protocol
RPC	Remote Procedure Call
CTI	Center for Testing and Interoperability
ETSI	European Telecommunications Standards Institute
TTP	Time Triggered Protocol
GMP	Group Membership Protocol
ETP	Event Triggered Protocol
EB	enhanced beacon
PAN	Personal Area Network
PCA	Priority Channel Access
ISM	Industrial, Scientific and medical
FHSS	Frequency-hopping spread spectrum
RPL	Routing over Low Power and Lossy Networks
CHMA	channel hopping multiple access
RTS	ready-to-send
CTS	clear-to-send
NF	noise floor
SES	exponential smoothing technique
RSSI	Received Signal Strength Indication
LQE	The Link Quality Estimation
ED	Energy Detection
LBCS	Load Balancing in the Cloud using Specialization
ROBUST	Reliable lOad Balancing Using Specialization for IoT critical application
WSANs	wireless sensor and actuator networks
LQE	Link quality Estimation
E-TSCH	Enhanced Time Slotted Channel Hoping

TSCH-CCA	Time Slotted Channel Hopping with Correct Collison Avoidance
I-LQE	Intelligent Link Quality Estimation
PDR	Packet Delivery Ratio
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
JADE	Java Agent DEvelopment Framework

General Introduction

The Internet of Things has two visions from both Internet-centric and Things-centric perspectives [1] [2]. The Internet centric architecture involves internet services where data is generated by Things. The Things centric architecture deals with the smart objects that take the center stage [1]. The Internet of Things (IoT) is an intelligent network. It basically integrates sensing, communications, and analytic capabilities in the communicating actuating networks. Sensors and actuators proliferate in the environment around us for the purpose of exchanging information and communicating through the information sensing devices [3] [4].

The IoT creates a myriad number of "Things" communicating each other. It utilizes the existing technologies and creates new communication modes [3]. It is able to deal with different technologies, such as Internet protocols, mobile communications networks, Wireless Sensor Networks (WSN), Radio-Frequency IDentification (RFID), and other sensing protocols [2]. It combines the virtual world and the physical world by innovating different concepts and components together: pervasive networks, miniaturization of devices, mobile communication, and new ecosystem. The miniaturization of devices stems from the fact that in recent technologies, computing power, storage, and battery capacities become available at relatively low cost and low size. This trend is enabling the development of extreme small-scale electronic devices with intelligent location, monitoring, tracking, identification, communication, computing and managing things capabilities, which could be embedded in other devices, systems, and facilities [3]. This evolutionary paradigm facilitates the human's daily life services. It cuts across many applications such as cities (smart lighting, air quality), smart metering (water flow, silo stock, water leakages), emergency (radiation levels, fire detection, access

control), retail (smart payment, item location), industry (process variable monitoring, safety), agriculture (green houses, smart irrigation), home automation (energy and water use, remote control), and e-healthcare (patient monitoring) [2]. On the hand, the miniature nature of these devices exposes the IoT applications to many issues. One of these issues is mainly related to the restricted physical capabilities (energy, processing, memory and the low data rate transmission) [5–7].

Wireless Sensor networks plays a vital role in IoT deployment since they are adaptable with the hard environments that a large number of WSNs can work independently of the Internet. Also, sensor nodes have features of mobility and heterogeneity. They provide more flexibility and lower cost to applications [8]. The disseminated sensor nodes are connected to each other via wireless communication and in a wireless multihope way to form a WSN [9]. The sensor node sends a site reference to the base station which in turn locates the region of occurred changes [8]. Some applications allow users to receive information about the temperature by using their smartphones to interact with sensor nodes placed in their region [9]. WSNs were firstly used in military applications, but due to the ease of sensors dissemination, their use has been generalize to other domains such as fire detection, Industries, machine, well being monitoring, intelligent homes, irrigation systems, intelligent transportation systems, E-health, and telematics [10], [11].

The creation of such a smart word, implies that all IoT devices to be connected to the internet in one form or another, which results in the generation of enormous amounts of data which have to be stored, processed and presented in a seamless, efficient, and easily interpretable form [12]. Cloud computing can provide the virtual infrastructure for such utility computing. Cloud computing services provide monitoring devices, storage devices, analytic tools, visualization platforms and client delivery. They allow users to access applications on demand from anywhere [1].

The inefficient storage of a huge number of data on Cloud computing servers can influence negatively on latency time, when servers are not load balanced. Cloud servers are very prone to failure, data redundancy is one of the optimum solutions to achieve data availability. However, achieving data availability is challenging when the sensed data is rapidly increasing and the Cloud servers are dynamic.

The proliferation of wireless communications has led to intensified utilization of the

2.4 GHz Industrial Scientific and Medical (ISM) band [13]. Wireless sensor networks (WSNs) in the 2.4 GHz presents many challenges due to the unreliability of the transmission medium along with the sensors miniaturization nature [14]. The most powerful equipment (e.g., Wi-Fi) contend with WSN for scarce spectrum which opens a big challenge that must be treated [13]. Wi-Fi and Bluetooth networks, and microwave ovens can cause high destructive interference in WSNs [15] [16] [17] [18]. In addition to the external interference problems, the harsh nature of the environments such as industrial environments can be crowded of metallic and mobile objects, such as robots, cars and people. These can expose WSN to large-scale and small-scale fading.

IEEE 802.15.4e proposed a TSCH mode dedicated to WSNs that increases the network throughput and addresses the different wireless technologies coexistence issues by providing periodical switch of the communication frequency. One of the limitations of TSCH stems from the fact all 16 IEEE 802.15.4 channels are used indiscriminately, whereas channels often experience different level of interference [13].

In our thesis, we develop an Internet centric approaches that treat the challenges of wireless sensor networks over IoT applications. We will address the energy deficit of sensors and packet delivery latency by reducing data retransmission. Data retransmission can occur from external and internal interference. Thus, we propose three contributions that achieve successful data retransmissions even in the existence of internal interference and external interference that comes from the harsh environments and the coexistence of the incompatible wireless technologies issues. Sensed data delivery time and data availability are also major challenges in IoT critical applications. We face such a problem, by eliminating collisions at the sensors level, and providing fast and intelligent storage on Cloud servers. We proposed three contributions to build a robust Infrastructure as a Service that achieves fast input and output critical requests to address the requests delivery time in critical applications, and achieves data availability within minimum latency and minimum resources wastage.

Dissertation Outline

The dissertation is organized as follows:

Chapter 1: Describes the Internet of Things, presents the IoT architecture, gives an overview of the essential elements that motivate the emergence of the IoT, discusses the research trends of the IoT and explores the quality of service criteria, the major challenges, the proposed solutions, and the future directions that must be addressed. The final section briefly concludes our work.

Chapter 2: Explains the problem statement of Cloud computing, and MAC-sublayer. It is divided into *two* main parts. The first part presents the state-of-the-art on load balancing. The second part provides a state-of-the-art on MAC-sublayer protocols, our classification of the protocols that achieved reliability when transmitting data, advantages and disadvantages.

Chapter 3: Introduces some IoT-applications that generate data that need to be stored on IoT-Cloud servers, proposed three contributions to build a new, fault tolerant IaaS for an intelligent storage in IoT-Cloud. The three contributions investigate data and data application's types to further improve Big Data structuring to address the bad storage, Input/Output response time challenges and achieve a fault tolerant and resilient system with minimum resources exploitation. They are based on parallelism, and collaboration of multi-agents system to achieve minimum latency. Our algorithm is implemented using JADE Platform(Java). The results showed significant improvements in terms of latency of critical data requests and critical data availability.

Chapter 4: Improves the reliability of the TSCH protocol by avoiding internal interference. To ensure internal collision avoidance in the presence of hidden nodes, this chapter presents two intelligent algorithms (Time Slotted Channel Hopping with Correct Collision Avoidance backoff algorithm(TSCH-CCA) and Enhanced Priority Chan-

nel Access Backoff Algorithm(E-PCA)) applied respectively to both normal packets and critical events packets to improve the latency, network congestion, network lifetime, critical event packets lifetime, and collision avoidance. This chapter also improves the reliability of communications of TSCH by applying two new, dynamic blacklisting techniques. The techniques select and use only the high-quality channels and blacklists the bad quality ones to overcome the negative effects of the co-existence of the different wireless technologies that share the same frequency. The proposed solutions has been simulated using ns-3 (Network simulator 3). The results show significant improvements in terms of throughput, energy and reliability.

PART ONE : THE STATE OF THE ART

Chapter 1: Challenges and Research Directions for Internet of Things

Chapter 2: Problem Statement and the Related Work

In recent years, the IoT has drawn significant attention as it can solve difficult problems. However, the heterogeneity of devices and the large scale networks expose the IoT to many challenges that must be addressed; otherwise, the systems performance will deteriorate. As an attempt to identify these challenges, this part comprehensibly reviews the main IoT concepts, the serious IoT challenges and the quality of services presented in the recent literature. It also investigates the corresponding main research directions and the proposed solutions.

Chapter 1

Challenges and Research Directions for Internet of Things

1.1 Introduction

The enormous growth of embedded devices along with the availability of the appropriate standard communication protocols have supported the building of the Internet of Things (IoT) systems. Wireless Sensor Networks (WSN), Radio-Frequency Identification (RFID), Internet protocols and mobile communications are the main communication protocols that enabled the emergence of IoT [2, 19]. The IoT refers to the next generation of the Internet [20]. It is also called the Internet of Everything [4].

The European Research Cluster on the Internet of Things - IERC [21] defines the IoT as *A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual 'things' have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network* [21]. It also asserts that *'Things' are active participants in business, information and social processes where they are enabled to interact and communicate among themselves and with the environment by exchanging data and information sensed about the environment, while reacting autonomously to the real/physical world events and influencing it by running processes that trigger actions and create services with or without direct man intervention* [21].

At Micrium [19], they define a 'Thing' as an embedded computing device (or em-

bedded system) that transmits and receives information over a network.

The embedded devices may contain a variety of interfaces that enable the system to manipulate, measure, and interact with the external environment (See Fig.1.1) [22]. Figure 1.1 illustrates an example of an embedded system that serves the fire protection of an environment. It periodically senses the humidity, the temperature, the carbon monoxide (CO), and the carbon dioxide (CO2) of the environment. When the CPU analyzes the captured data and if it detects abnormal events, it applies the necessary countermeasures to put out the fire.

— The human interface may be simple (flashing light) or complicated (real-time robotic vision) [22].

— The diagnostic port may be used for diagnosing the system [22].

— Application specific integrated circuit (ASIC), special-purpose field programmable gate array (FPGA) may be used to increase the safety or performance [22].

— Software often has a fixed function. It is specific to the application [22].

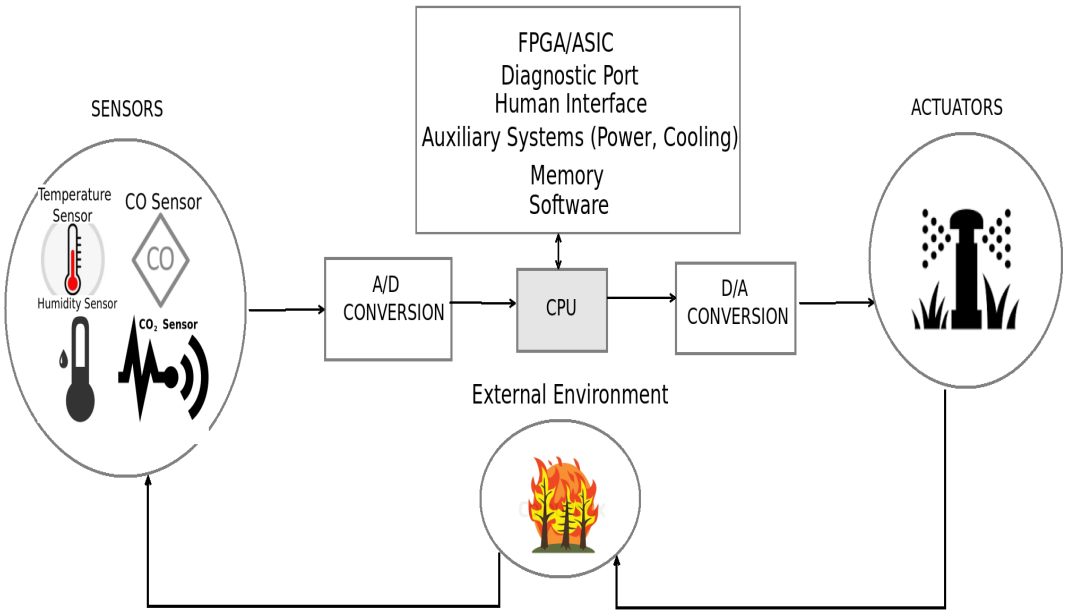

Figure 1.1: An Example of an embedded system

It is mentioned in [19] that the embedded systems are based on microcontrollers (MCUs) and run software with a small memory footprint.

The Ipv6 motivated the emergence of the IoT since it enables the unique identification of all the objects. Hence, the Radio Frequency IDentification (RFID) group [23]

defines the IoT as *the worldwide network of interconnected objects uniquely addressable based on standard communication protocols.*

The Micrium members [19] categories the IoT system into 2 main classes: Industrial IoT and Commercial IoT. The devices of the Industrial IoT have the ability to be connected to an IP network (network layer). Unlike the Industrial IoT devices, the Commercial IoT devices can only communicate with local devices via Bluetooth or Ethernet (the physical and data link layers, LANs) wired or wireless [19]. They illustrate that the IoT is composed of four main components: the objects, the local network (gateway), the Internet and back-end services (enterprise data systems, or PCs and mobile devices) (See Fig. 1.2) [19].

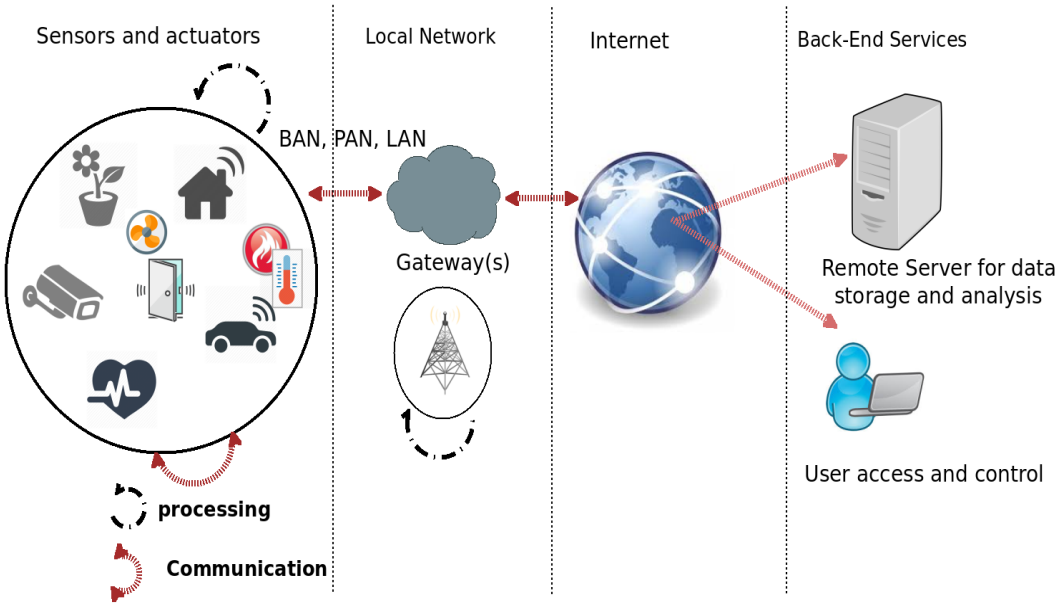

Figure 1.2: The IoT from an embedded systems point of view.

The International Telecommunication Union (ITU) [24] defines the Internet of things as *a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies. NOTE 1 - Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled. NOTE 2 - From a broader perspective, the IoT can be perceived as a vision with technological and societal impli-*

cations.

Guabbi *et al.* [1] suggested that the IoT is the *Interconnection of sensing and actuating devices providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless ubiquitous sensing, data analytics and information representation with Cloud Computing as the unifying framework.*

S.Hammoudi *et al.* [5] define IoT as follows: The Internet of Things is a set of heterogeneous connected devices that collaborate with the Internet of services via the communication protocols. The communication protocols are importantly dedicated to the IoT devices' constraints. Mainly, this collaboration aims to offer services needed by the end user.

The IoT's services have the full potential to cover diverse domains. In recent years, the IoT has attracted a growing amount of interest, due to its numerous and attractive services. These services are able to cover diverse domains, such as cities, smart metering, emergency, retail, industry, agriculture, home automation, and e-healthcare [2, 4]. The IoT devices have the ability to be used in every single physical object, such as clothes, houses, buildings, campuses, factories, and human bodies [19]. These uses, along with the communication protocols, created a totally different environment with smart objects. The smart devices can predict and understand what we need and how to react to any kind of stimuli. They also have the ability to communicate with users and with each other. To achieve their tasks, they can share data and services [25]. The creation of such a smart environment requires the deployment of devices in high volumes. The CISCO Internet Business Solutions Group (IBSG) [26] estimated the number of IoT devices in the near future, to be 50 billion by 2020.

Despite the growth potential of the IoT devices and the high emergence of the interesting services that will be offered, the IoT faces many issues. These issues are mainly related to the addressing, routing, standardization and the restricted physical capabilities (energy, processing, and memory). The architecture is also one of the main issues since it must support a large number of the heterogeneous devices and manage the application's dependencies. As the IoT networks are deployed on a large scale, security,

privacy, and openness are considered as a major concern that inevitably affects the IoT. To build an efficient system, availability, reliability, management, interoperability, and robustness must be ensured. These points are also considered among the IoT specific challenges.

This chapter is organized as follows [5]. In section II we present the IoT architecture. In section III, we overview the essential elements that motivate the emergence of the IoT. Section IV discusses the research trends of the IoT. Section V explores the quality of service criteria, the major challenges, the proposed solutions, and the future directions that must be addressed. The final section briefly concludes our work.

1.2 Internet of Things Architecture

Objects in the IoT network must be inter-connected. The extensibility, scalability, and interoperability among heterogeneous devices must be verified in the IoT network. Also, the device's physical characteristics, such as the ability of moving geographically and the real-time communication between devices must be considered while designing the IoT's architecture [20]. This section presents the important layers of Internet of Things architecture.

1.2.1 Sensing layer

In the sensing layer, the smart systems capture the environmental information and exchange data among devices in a persistent and automatic way. To control and track every single object deployed in such large scale systems, each device must have a unique digital identity. The Universal Unique IDentifier (UUID) is the technique of assigning a unique identity to each object [20]. There are many important aspects that should be taken into consideration while determining the sensing layer of an IoT, such as cost, size, resource, energy consumption, the manner of the devices deployment, the heterogeneity, the network topology, the communication, and the protocols [20]. The main techniques in this layer are the sensing technology, RFID technology, 2-dimensional (2-D) barcode, GPS, etc. [27].

1.2.2 Network layer

It is necessary for a network to automatically determine and map things in the network. The network layer in the IoT is able to connect all things that can share and exchange data. Thanks to this layer, all things are always informed about their surroundings. This layer is also responsible for transferring the information from the sensing layer to the service layer [27]. To ensure a reliable service, the Quality of Service (QoS) should be accounted for the communication in the network [20]. The wireless technologies used for data transmission can be: 3G, Bluetooth, infrared, ZigBee, wifi, etc. [27].

1.2.3 Service layer

The service layer is based on the middleware technology which supports the services and the applications in the IoT. The middleware also ensures the interoperability among the heterogeneous devices [28]. The service layer performs many useful services, such as information search engines and communication, exchanging and storage, management of data, and the ontology database. To locate services for any application and to dynamically retrieve metadata about the services, the services in the service layer are executed directly on the network. A practical service layer is composed of a set of common requirements of applications, application programming interfaces (APIs), and protocols aiding demanded applications and services [20]. The exacerbation of the number of sensors and the huge information they sense are addressed by the Cloud computing technology. Cloud computing is the key technology in this layer since it has the full potential to store and process big data [27].

1.2.4 Interface layer

To realize a clear and comprehensible interaction among the heterogeneous things, the compatibility issue must be addressed. To make the management and the interconnection of things easy, an effective interface mechanism must be used. Interface Profile (IFP) can be considered as a subset of service standards that allows minimal interaction with the applications running on the application layers. For instance, Universal Plug and Play (UPnP) specifies a protocol for the seamless interactions among

heterogeneous things [20].

1.3 Elements of the Internet of Things

1.3.1 Wireless Sensor Network (WSN)

The low-power integrated circuits and the wireless communications motivate the creation of the miniature devices that can be used in remote sensing applications [1]. The WSN is composed of a set of physical autonomous sensors (hundreds or even thousands) that are spatially distributed in the environment. The utility of these physical components is monitoring the physical or the environmental conditions, such as temperature, sound, pressure, etc. [12, 19]. The WSNs are heavily used in logistics where the transported products are temperature-sensitive. They are also used in tracking and maintenance systems as they have the ability of real-time data analysis [4]. Mostly, sensors are connected to one another, the captured data of the sensor is passed from one node to another through the network. They cooperate together to send the data to a central location [12, 19]. The fundamental components that make up the WSN monitoring network include:

— **WSN hardware:** This is the physical part of a device. It is composed of sensor interfaces, processing units, transceiver units, a power supply, and the analog and digital (A/D) converters [1].

— **WSN communication stack:** A tight design of the topology and routing mechanisms are quite important for the longevity of the network. Typically, nodes in the WSN communicate with each other to transmit data to a base station. The sink node should play the role of a gateway so that it can interact easily with the Internet [1].

— **WSN middleware:** The collaboration of cyber infrastructure with a Service Oriented Architecture (SOA) and sensor networks can allow access to heterogeneous sensor resources. For the development of sensor applications to be feasible, a platform-independent middleware must take place [1].

— **Secure data aggregation:** To ensure that the collected data from the sensors are trustworthy and, to keep the network working in a reliable and persistent way,

a data aggregation method must be put in place. Failure detection and recovery are required in case of node failure as well as security from malware is a critical factor that must be guaranteed [1].

1.3.2 Radio Frequency IDentification (RFID)

Radio Frequency IDentification (RFID) is a technology that facilitates the automatic tracking and identification of tags attached to any object or person [1, 12]. The RFID tag can warehouse much more data than a bar code which identifies only a category of product. In other words, a mini-database is hosted in the item. The majority of applications define the read range of RFID tags by one meter. Other applications set the limit between $0.15--0.20$ meters. Newer tags in the Ultra High Frequency (UHF) radio frequency bands are able to have a range of 6.0–7.5 meters [12]. The main three types of tags are:

— **Passive RFID:** This type of tag is not battery powered. It is supplied with energy from radio frequency energy sent by the reader [1, 4].

— **Active RFID:** The active RFID is supplied with batteries that enables it to launch communication with a reader. Also, it has external sensors to monitor temperature, pressure, chemicals etc. [1, 4].

— **Semi-passive RFID:** This tag has an internal battery to power the microchip. It also derives the power from the reader while communicating to supply its power [4].

The WSN can monitor the physical entities as well as our environment. It can collaborate with RFID systems to maintain better control (locations, temperature, and movements) [4].

1.3.3 Middleware

Middleware is a software layer mediated between software applications [4]. It is intended to facilitate and simplify the establishment of the communication as well as the development of new services in the distributed computing environment. These benefits aid the development of the IoT applications with many heterogeneous devices which are disseminated in a complex distributed infrastructure. For instance: Global

Sensor Networks (GSN) are an open source sensor middleware platform enabling the development of sensor services [4]. In [29], GSN is based on a virtual sensor. Rapid deployment is supported by deploying the specification of virtual sensors provided in a declarative deployment descriptor. The specification of virtual sensors provides all necessary information required for deploying and using them. The GSN is built based on four characteristics: simplicity, adaptivity, scalability, and light-weight implementation [29].

1.3.4 Protocols and addressing schemes

The unique identification of every single object in the world has great importance. It makes the control of the remote devices through the Internet easier. However, IPv4 is not capable of addressing a huge number of devices which leads to the use of IPv6. Another critical development in addressing devices is a lightweight IPv6 [1]. In the IoT, TCP/IP guarantees a reliable routing from the source to the destination. The common problem have been faced by the IoT is the bottleneck problem at the interface between the gateway and wireless sensor devices. Furthermore, it is required that any new deployed devices or any new networks must not deteriorate the performance, robustness, and trustworthiness of the network. Because of the importance of the device address and to be able to solve all these issues, the Uniform Resource Name (URN) is considered as a crucial point for the development of the IoT [1]. Figure 1.3 illustrates the IP Smart Objects Protocol stack.

1.3.4.1 Constrained Application Protocol (CoAP)

The IETF Constrained RESTful Environments (CoRE) working group produced the Constrained Application Protocol (CoAP). It is an application layer protocol created for constrained nodes, which are characterized by intensive limits on energy, throughput, and storage, like WSN nodes that build the IoT [30]. The CoAP is a web transfer protocol based on REpresentational State Transfer (REST) on top of Hypertext Transfer Protocol(HTTP) functionalities [25,31]. It is intended to achieve its goals with less complexity. The CoRE group has proposed the following features for CoAP: RESTful protocol design minimizing the complexity of mapping with HTTP, low header overhead and parsing complexity, Uniform Resource Identifier (URI) and content-type

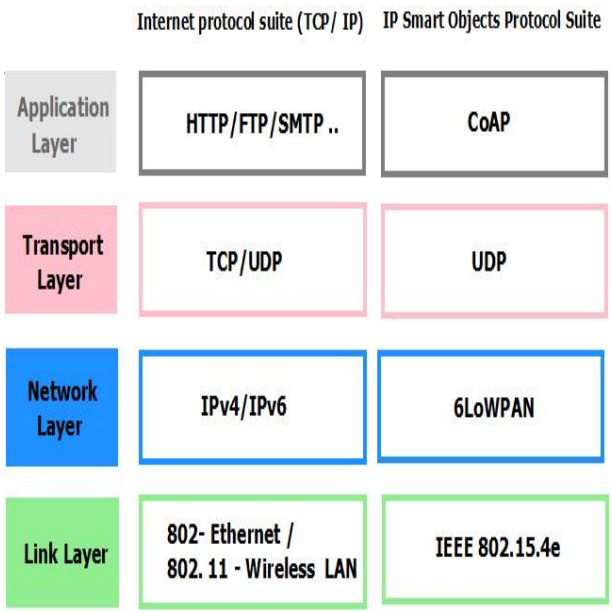

Figure 1.3: IP Smart Objects Protocol stack.

support, support for the discovery of resources provided by known CoAP services, simple subscription for a resource, and resulting push notifications, and caching based on max-age [31].

REST exchanges data in a lightweight manner between clients and servers over HTTP. Unlike SOAP, REST does not use XML to make requests, it simply relies on the URL. To perform the required tasks and to manipulate with the web services, REST uses four different verbs (GET, POST, PUT, and DELETE) and uses Uniform Resource Identifiers (URIs) as nouns. It presents a cacheable connection protocol with a stateless client-server architecture. It is used within both social and mobile network applications [25].

The CoAP handles two sub-layers: the Transaction sub-layer and the Request/ Response sub-layer [32], as illustrated in Figure 1.4.

The transaction sub-layer treats the single message delivered between the communicating nodes. It provides reliable communication over the User Datagram Protocol (UDP) transport layer using exponential backoff. It also detects message duplications. This sub-layer presents four types of messages [25, 32].

— **Confirmable:** a message that obliges an acknowledgment.

— **Non-conformable:** a message that does not need to be acknowledged.

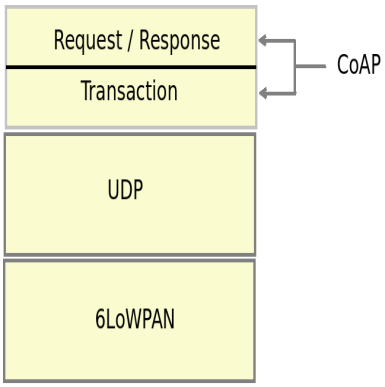

Figure 1.4: The CoAP protocol stack.

— **Acknowledgment:** it acknowledges a Confirmable message.

— **Rest:** indicates that a confirmable message has been received but context is missing to be processed.

The ***request/response*** sub-layer handles REST communications. Reliability of the CoAP is achieved by both confirmable and non-confirmable messages.

The HTTP suffers from large message overhead, which implies packet fragmentation. This problem significantly deteriorates the performance of the Low power and Lossy Networks LLNs. The main aim of the CoAP is to reduce the overhead as much as possible and limit the use of the fragmentation. A typical request has a total header of about 10−−20 bytes. It applies the header compression mechanisms to achieve its goals [25]. The CoAP packet format is illustrated in Figure 1.5 [25, 32].

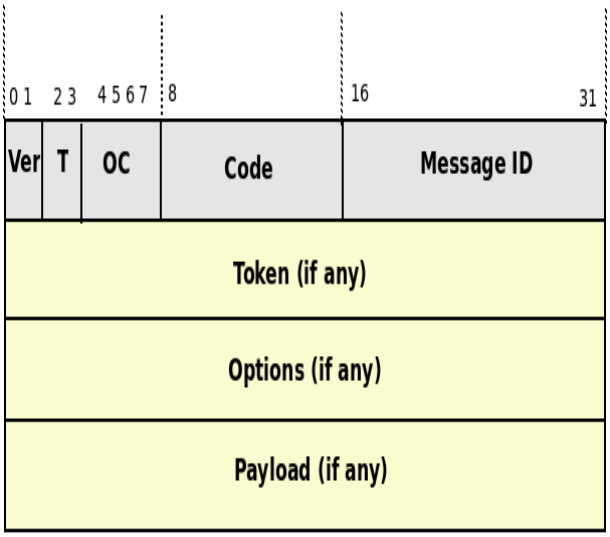

Figure 1.5: CoAP packet format.

The token value is intended to correlate the requests and the responses. Its length ranges from zero to eight bytes [25].

The options and payload are the next optional fields [25].

The fields in the header are as follows [25]:

— **Ver** is the version of CoAP,

— **T** is the type of transaction,

— **OC** is the option count,

— **Code** represents the request method (1−−10) or response code (40−−255). For example, the code for GET, POST, PUT, and DELETE is 1, 2, 3, and 4 respectively [25].

— **Transaction ID** is a unique identifier for correlating the response.

Colliti *et al.* [32] analyzed a client-server transaction and quantified the amount of transferred bytes and packets when using CoAP and HTTP. The results show that the number of bytes per transaction is 128 in HTTP and 1,288 in CoAP. The number of packets per transaction is 17 in CoAP and 2 in HTTP.

1.3.4.2 6LoWPAN

The adoption of IPv6 on top of a low power Wireless Personal Area Network (WPAN) is inapplicable since the low power WPANs has low bandwidth, small packet sizes, and battery supplied devices with limited processing capabilities. The IETF IPv6 over Low power WPAN (6LoWPAN) working group started in 2007 to work on specifications to allow the IPv6 packets to be transported over IEEE 802.15.4 MAC. They define the encapsulation and the header compression to reduce the overhead, the fragmentation to satisfy the IPv6 Maximum Transmission Unit (MTU) requirements, and mesh networking that supports layer-two forwarding [25, 31–33].

6LoWPAN expresses each capability in a self-contained sub-header (See Fig. 1.6). Each sub-header is identified by two bits [25, 32, 33].

***(00)* NO 6LoWPAN Header:** used to specify the non compliant packets to the 6LoWPAN specification. These packets will be discarded.

***(01)* Dispatch Header:** responsible for the management of the link-layer, multicasting and the compression of the IPv6 headers.

(10) Mesh Addressing: used for the layer-two forwarding.

(11) Fragmentation: headers used when datagram lengths exceed a single IEEE 802.15.4 frame.

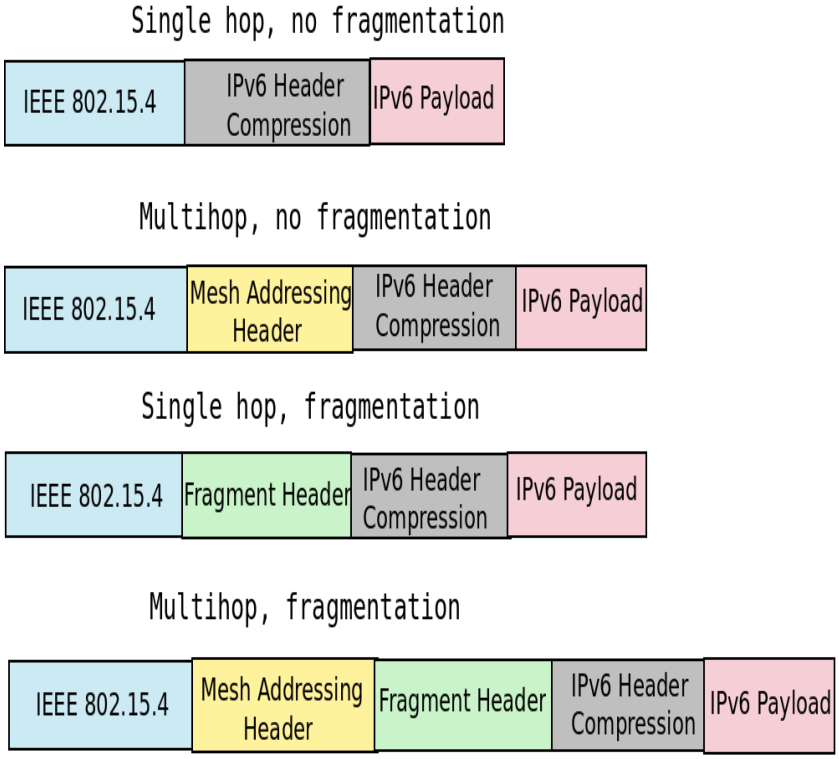

Figure 1.6: 6LoWPANs headers.

1.3.4.3 IEEE 802.15.4e

The IEEE 802.15 working group created the IEEE 802.15.4 standard, which specifies the sub-layer for Medium Access Control (MAC) and a physical layer (PHY) for Personal Area Networks (PANs). It also focuses on low power consumption and low cost devices. It provides reliable communication with high message throughput. It can operate on different platforms. It realizes high level authentication, encryption, and security services [34].

The IEEE 802.15.4 sensors are divided into three categories: one Coordinator, Full Function Devices (FFD), and Reduced Function Devices (RFD). The coordinator is responsible for creation, control, and maintenance of the network. It can store a routing

table within its memory. The FFD has the capacity to serve as a personal area network (PAN) coordinator [34, 35].

The FFD has two roles: sense data and route packets to other devices. The RFD is a device with limited resources. It senses data and sends it to another FFD or to the coordinator. The FFD forwards the packets into another FFD until the packets reach the coordinator [34, 35].

The IEEE 802.15.4 standard allows use of three topologies which are: star (single-hop), cluster-tree, and mesh (multi-hop) [25, 34, 35] (See Fig. 1.7).

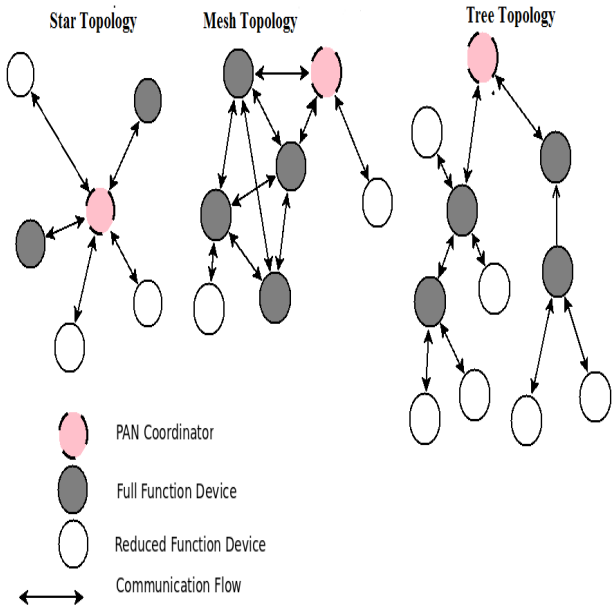

Figure 1.7: IEEE 802.15.4 topologies.

The IEEE 802.15.4 e presents four modes: Time slotted channel hopping (TSCH), Deterministic and Synchronous Multi-channel Extension (DSME), Asynchronous multi-channel adaptation (AMCA), and Low Latency Deterministic Network (LLDN).

— **Time Slotted Channel Hopping (TSCH):** TSCH combines the time slotted access, the multi-channel, and channel hopping techniques. These techniques aid to eliminate the collision problems, provide deterministic latency to applications, and increase the network's capacity. This mode achieves reliable communications as it uses the channel hopping technique. The channel hopping technique reduces multi-path fading and interference. The TSCH can be used to form any network topology (e.g., star, tree, partial, or full mesh) [34, 36].

— **Deterministic and Synchronous Multi-channel Extension (DSME):** To

guarantee robustness and high reliability, the DSME selects the best communication channels by using the channel diversity. This mode is designed for critical applications that require a deterministic delay. Technically, it combines contention-based and time-division medium access. It is designed for mesh and multi hop networks [34, 36].

— **Low Latency Deterministic Network (LLDN):** The LLDN protocol is dedicated to factory automation applications that require lower latency. The nodes in this mode use a single-hop and a single-channel to communicate. The frequency channel is defined by the PAN coordinator during the network formation [34,36].

— **Radio Frequency Identification Blink (BLINK):** The BLINK mode is used for item/person identification, location, and tracking. The ID's node communicates without using prior association, or using acknowledgement. The BLINKS packets are transmitted using the Aloha protocol [34].

— **Asynchronous multi-channel adaptation (AMCA):** This protocol is dedicated to application domains with large deployment. The AMCA uses asynchronous multi-channel adaptation. The device chooses the best quality channel to transmit its data [34].

Table 1.1 classifies these modes according to their application domains.

1.3.5 Data storage and analytics

It is required to store the data on centralized data centers based on harvesting energy techniques so that we ensure energy efficiency and reliability. Also, artificial intelligence algorithms are needed to assure the validity of the gathered data as well as the automated decision making. The development of a centralized infrastructure that supports storage and analytics is important. Cloud-based storage services solutions are becoming well known and Cloud-based analytics and visualization platforms are under development to be more prevalent [1].

Timothy *et al.* [37] presented the Water-flow Visualization Enhancement (WaVE). The WaVE has an extensible and flexible framework and toolset. It integrates enhanced geospatial analytics visualization and decision support modular tools. It has the potential to transform historic, real-time, and forecasted stream flow and flood inundation data into accurate actionable intelligence.

Table 1.1: The IEEE 802.15.4e protocols' classification

IEEE 802.15.4e modes	**Application's domains**
TSCH	process automation applications Equipment and process monitoring oil and gas industry, food and beverage products, chemical products, pharmaceutical products, water/waste water treatments, green energy production, climate control.
LLDN	factory automation where applications require very low latency home automation, smart metering, smart buildings and patient monitoring
BLINK	item/people identification, location, and tracking
AMCA	smart utility networks, infrastructure monitoring networks, process control networks

1.3.6 Visualization

To realize a successful interaction of the user with the environment and to convert data into knowledge, an easy, clear, and comprehensible presentation of the information is required. So, the visualization is a critical point in the IoT [1].

PubNub is a global Data Stream Network (DSN) and real-time infrastructure-as-a-service. The pubNub created the EON project: Open Source Framework for Realtime Dashboards and Maps of the IoT applications. The EON is an open source JavaScript framework for mapping and charting real-time data. The EON efficiently reads the values captured by sensors (e.g., attached to Arduino) and plots a graph on the web [38].

The 7" Touchscreen Display for Raspberry Pi enables users to create all-in-one, integrated embedded projects [39].

Open Geospatial Consortium(OGC) Sensor Web Enablement (SWE) enables the accessibility and the usability of all types of sensors, and sensor data repositories to be discoverable. Developers can achieve such a benefit via the web. The Sensor Observation Service (SOS) is one of the main OGC standards in the SWE framework, where SOS is an open interface for a web service to obtain observations and platform descriptions from one or more sensors [40].

1.3.7 Cloud computing

The vision of the IoT can be seen from two perspectives 'Internet' centric and 'Thing' centric. The Internet centric architecture will involve internet services being the main focus, such as Cloud computing while data is contributed by the objects. In the Object center architecture, the smart Things take centre stage [1].

The National Institute of Standard and Technologies (NIST) defines the Cloud computing as follows *'Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned (supplied) and released with minimal management effort or service provider interaction'* [41]. Gmail, Social networking sites that share photos and videos (e.g., Facebook), online file storage, and online business applications are some commonly prevalent provided Cloud services [42, 43]. Cloud computing is based on virtualization and distributed computing, where the end user can exploit the hosted services without knowing any details about the backend used resources [43]. Cloud computing plays a crucial role in the emergence of the IoT. It addressed some serious IoT issues as it offers unlimited storage resources and processing power, whilst most of the IoT devices are miniature devices (with limited storage processing capacity, and power) [2, 41]. CloudIoT is the commonly used term that expresses the combination of the two complementary technologies: IoT and Cloud computing. Cloud computing plays a vital role in the emergence of new smart services and applications that can highly influence our everyday life. It also empowers the evolution of some needed applications in our daily life, such as smart home and smart metering, video surveillance, smart energy and smart grid, smart logistics, environmental monitoring, and health care services [41].

1.3.7.1 Cloud computing components

Amin *et al.* [43] classified the components of the Cloud computing into three main components: clients, distributed servers, and data centers.

Clients: represent the end users' terminal that they use to manage the information on the Cloud. Typically, the terminal can be a laptop, mobile phone, PAD, or others [42, 43].

Data centers: are a set of servers that host the required services. Virtualization

is used to create virtual servers on just one single physical server in a data center [43].

Distributed servers: are the physical servers situated in different areas. They offer better accessibility and guarantee the security [43].

1.3.7.2 Cloud computing service models

Cloud computing can offer three different types of services (see Fig. 1.8):

Software as a Service (SaaS): Complete applications are hosted at the backend rather than being hosted in the end users' terminal or in a local data center. The supplied applications on demand to the user are running on Cloud environments and accessed over the Internet through a client or a web browser. Many SaaS applications are free to use. They are also collaborative, which means that multiple end users can share the same services and documents at the same time [41–43].

Platform as a Service (PaaS): A platform is a software environment used to develop and run applications [42]. PaaS offers developers online access to platform-layer resources (e.g., operating system support, software development frameworks, application servers) hosted in the Cloud in order to build their SaaS applications. The developed applications using the PaaS service can be offered free for anybody on the web as they can be private to one or a few persons within a company [41–43]. Developers are restricted by the tools offered by the Cloud vendor, so their applications can be developed employing just the offered services [42].

Infrastructure as a Service (IaaS): In the 'IaaS', the users are allowed to store their data, develop and run their applications. The 'IaaS' supplies processing, storage, and network resources. At the same time, it allows the clients to control the storage and the applications. The servers of Cloud computing where the work will be affected can be either physical or virtual [41–44].

1.4 Research trends of the IoT

During the last 10 years, Google research trends have been measuring the number of times that the IoT key has been checked in Google navigator. They clearly noticed that the IoT research is constantly increasing [1]. The IoT is also expected to take 5-10 years to be adopted by the market [1]. Technically, the trend of the IoT is the

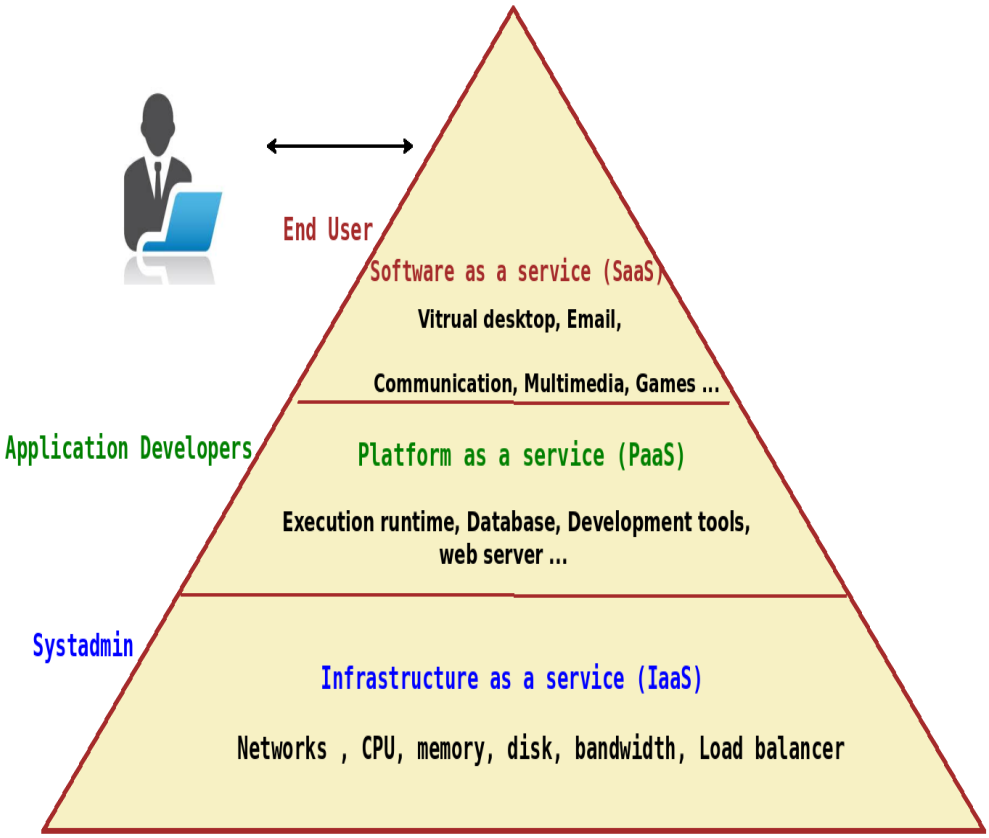

Figure 1.8: Cloud computing architecture.

incorporation of sensing and the Internet. All the networked things should be flexible, smart, and autonomous enough to provide required services. It will provide our daily lives services with desired connectivity and intelligence [20].

1.5 QoS Criteria, IoT Challenges and Future Directions

1.5.1 Architecture and dependencies

In the presence of a huge number of objects connected to the Internet, it is indispensable to have an adequate architecture that supports an easy and large connectivity, tight control, ideal communication, and useful applications [45]. Designing a SOA for the IoT is a big challenge for a large scale network [20]. The automatic service composition based on the requirements of applications is still a challenge as well [20]. To realize the services' compatibility in different implementation environments, a commonly accepted service description language is required. Also, a robust

service discovery is essential to progress the IoT technology [20]. Since the app store can offer novel and unbounded development applications that can simply be executed on smart phones, they may be exploited and used as an architectural approach for the IoT [45]. Guabbi *et al.* [1] present an architecture that suits some IoT application domains. It is based on Cloud computing, where cost based services are required. The European Union projects of SENSE and the Internet of Things-Architecture (IoT-A) have been basically focusing on challenges from the WSN aspect for being the prevalent component in the IoT [1]. Generally, an overall architecture needs to be tightly developed according to the needs and functionalities of fields so as it can supply the necessary network and information management services to empower a reliable and accurate context information retrieval and actuation on the physical environment [1].

It is noted that the IoT systems are not complicated, but designing and building them can be a complicated task [19]. Different applications with shared sensor and actuator devices may have to run simultaneously to build a complete, reliable, and synchronized system [45]. If the system is not tightly and carefully designed, systems of systems interference problems may occur [4,45]. In a hyper-connected world, if an error occurs in just one part of a system, this can cause a dislocation in the whole system, which may turn the human life into chaos [4]. To alleviate such a serious problem, a comprehensive approach for specifying, detecting, and resolving dependencies across applications should be considered while designing the IoT architecture [45]. Also, in hyper-connected IoT systems, a reduction of the complexity of connected systems, the enhancement of the security, standardization of applications, and the guarantee of the safety and privacy of users must be available anytime, anywhere, and on any device [4].

1.5.2 Security

The frequent major problem that can really negatively impact the network systems is security attacks. Security attacks can easily affect the IoT systems for many reasons, such as the minimal capacity of the smart devices, the wireless communication between devices, the openness of systems, and the physical accessibility to sensors, actuators, and objects [45]. Hence, RFID, WSN and Clouds are the three IoT hardware that might be subjected to such attacks. It is noticed that RFID is the most vulnerable device as it identifies and tracks persons as well as objects [1, 23]. Devices are notoriously

prone to failure. Once a device is broken, it can be easily exploited by attackers. In this case, the system with high fault tolerance must be deployed. The redundancy of devices enables the system to continue operating properly even when there is a device failure [45]. In [4], it is mentioned that 70% of the IoT devices are vulnerable and are prone to any kind of malware. This problem is prevailing on account of the lack of transport encryption, insecure Web interfaces, inadequate software protection, and insufficient authorization [4]. To have a reliable system, its data must not be altered [1]. The nodes in the system must be authorized to send or receive data streams [46]. The message authentication codes guarantee the integrity and authenticity [45]. So, the authentication is a critical and important factor that the system must ensure. The authentication problem can be found in the traditional network reprogramming, which just consists of a data dissemination protocol that distributes code to all the nodes in the network without authentication, and this is a threat that affects security [1]. To achieve a secure programming protocol, the device must be aware about the authenticity of every code to prevent any malicious installation [1].

To address the security attack problems and to let the IoT system work properly, the IoT system needs to detect the attack, diagnose it, and deploy countermeasures and repairs [25]. In other words, it must also adapt to new, unexpected attacks, especially when the system is first deployed [45]. As most of the IoT's devices are resource-constrained devices [46], these procedures must be performed in a lightweight manner [45]. To have a reliable system, all these procedures must be performed using real-time computation [45]. Procedures that train developers to integrate security solutions (e.g., intrusion prevention systems, firewalls) are needed [4].

To gain the social acceptance of the IoT technologies, the trustworthiness of information and protection of private data must be tightly verified [20]. The data provenance and integrity, identity management, trust management, and privacy are the main challenges in building a secure IoT architecture [46]. These points are so critical as data provenance ensures that the source of data is trustworthy, which refers to the trustworthiness of the system. Data integrity ensures the absence of unauthorized alteration of the data. Trust management ensures trust in the devices or embedded systems. Identity management refers to the administration of individual identities [46]. Kanuparthi *et al.* [46] outlined the main different attacks that can affect these four main challenges.

They proposed appropriate solutions for each attack.

— **Challenge 1: Data Provenance and Integrity:** The first attack that can affect the data provenance and data integrity is when the physical sensor is maliciously modified to send incorrect values. The sensor Physical Unclonable Functions (PUF) is considered as a perfect solution as it provides authentication, unclonability, and verification of a sensed value [46].

— **Challenge 2: Identity Management:** A malicious node can falsify its identity into a genuine one to be authorized to send data, to the node/ sensor or the actuator to carry out some unnecessary actions. The proposed solution for such problem is sensor PUFs. The sensor PUFs have the ability to supply unique IDs. A lightweight identity management application that performs the necessary tasks is required [46].

— **Challenge 3: Trust Management:** The applications of the IoT's devices must be protected against attackers. Also, the actuators must be protected against the inputs generated from unauthorized sensors. The attackers can affect the application by secretly executing a rootkit. The proposed solution for this case is the implementation of the Hardware Performance Counters (HPCs). The HPCs are registers that can monitor certain events that occur during the lifetime of a program so that they can detect the integration of the rootkit program [46].

1.5.3 Privacy

The mobile crowd sensing and cyber-physical Cloud computing IoT technologies contribute to the emergence of the smart and connected communities (SCC). The goal of building SCC is to live in the present, plan for the future, and remember the past. Since SCC are human-centered systems, security and privacy are considered as a major concern that must be seriously addressed. For example, in order to solve traffic problems, GPS sensor readings can be used to estimate traffic congestion. On the other hand, they can be also used to show private information about people (e.g. home, work locations , etc.,) [47].

The privacy problem can occur when two devices communicate with each other. The attacker can recover the private information transformed from the sender to the receiver or vice versa [46]. As the cryptographic methods are intended to protect the system

against outside attackers and ensure data confidentiality [45], lightweight encryption algorithms are the ideal solution to guard against the eavesdropping attack and to ensure the confidentiality of the data [46]. Also, the existing encryption technology used in WSNs can be extended and deployed in the IoT [20], such as encryption algorithms, hash functions, digital signatures, and key exchange algorithms [46]. The selection of the appropriate algorithm should be according to the IoT's capacity constraint like power and memory [46]. The possibility of the interconnection of a myriad number of devices in the IoT system is also a reason that violates the privacy [45]. To respect the devices' privacy, the definition privacy from the social, legal, and cultural perspectives is needed [20]. Also, a set of dynamic privacy policies must be deployed [45]. In the case of an incoming user request, an evaluation of the request must be applied. If the users' request respect the established policies, the request will be granted; otherwise, it will be denied. In this case, the IoT paradigm must be able to express users' requests for data access and the policies [45]. The main issue is the need for the creation of a new language that expresses privacy policies. This lack is caused by the difficulty of the interpretation of many requirements [45].

1.5.4 Openness

Openness matters a lot in such a broad scale network as it makes it easy to remotely control devices by sending data over the Internet. Another great benefit is the cooperation and 2-way control. It is noticed that the benefits cannot be achieved except in the presence of the open communication systems, where several objects can constantly communicate with each other [45]. On the other hand, openness creates many new research problems that must be considered while designing the system, such as security and privacy [45, 46]. Feedback control is also one of the problems that openness encounters. Usually sensors and actuators use feedback control theory to provide robust performance. Studies have proven that stochastic control, robust control, distributed control, and adaptive control are not developed well enough, so a rich set of techniques must be employed [45].

1.5.5 Standardization

Standardization is considered as a main part that aids the development of IoT. However, rapid emergence of IoT makes the standardization difficult. The common issues of the IoT standardization are: radio access level issues, semantic interoperability and security, and privacy issues [20]. Moreover, APIs are also quite demanded in the interconnection among heterogeneous smart objects. The web based interfaces are considered as the suitable solution that facilitates these communications, especially with the IoT devices and Cloud computing while transmitting data for storage [41]. The main issue resides in the absence of the specific design for an efficient machine-to-machine communication [41]. The open standards of the IoT, such as security standards, communication standards, and identification standards, might empower the growth of the IoT technologies [20].

1.5.6 New protocols

The protocols play a main role to ensure the exchange data between devices and the efficient transport of embedded devices' data to the Internet. In familiar web protocols and in several MAC protocols there have been many proposed protocols for various domains with Time Division Multiple Access(TDMA), Carrier Sense Multiple Access (CSMA) and Frequency Division Multiple Access (FDMA). Although, all these protocols cannot suit the IoT devices well. So, to achieve acceptable results, new lightweight protocols must be redefined taking into account the constraints of the device's energy. Sensors are vulnerable, therefore they can be out of order at any moment and for any reason. To build a reliable system, the latter must be capable to perform as a self-adapting network and to allow for multi-path routing. The number of hops in the multi-hop scenario will be limited. For that, routing protocols must be modified to take into consideration the energy constraints of the device [1].

1.5.7 Energy

The huge number of heterogeneous simultaneous sensing of the urban environment negatively affects the network traffic, data storage, and energy utilization. The reduction of energy consumption is a critical point due to the limited power of the IoT

devices. So, a set of techniques that are capable to improve energy efficiency are required [1]. Compressive sensing enables one to reduce the signal measurements without having an effect on the accurate reconstruction of the signal [1]. Using sampling methods, the system can lower the data transmission rate [48]. Also, the transmission power of each sensor can be reduced due to the synchronous communication used by compressive wireless sensing (CWS) [1].

1.5.8 Extracting knowledge from big data

In the IoT, the quantity of the generated data from the myriad number of smart objects disseminated on planet earth is growing exponentially [49]. It is about streaming data that gives information about location, movement, vibration, temperature, humidity, and chemical changes in the air, etc. [4]. In the era of IoT, the SCC deploys Wireless Sensors that may cover diverse application domains such as, healthcare, environmental, smart buildings, and smart interconnected automobiles and trucks. The main problem is how to translate physical, biological, or social variables into a meaningful electrical signal [47]. To handle the data heterogeneity, SUN *et al* [47] proposed to address the following issues: How to improve quality (accuracy) of data in real time; how to improve intelligent data interpretation and semantic interoperability; how to unify data representation and processing models to accommodate heterogeneous or new types of data; how to implement inter-situation analysis and prediction; how to implement knowledge creation and reasoning; and how to conduct short-term and long-term storage.

Big data stream mobile computing (BDSMC) is a new paradigm emerged from the convergence of broadband mobile Internet networking (B-MINet) and real-time mobile Cloud computing (Rt-MCC). The BDSMC aims at describing a new generation of mobile/ wireless integrated computing-networking infrastructures [50]. Baccarelli *et al* [50] proposed "five Vs" formal characterization of the BDSMC paradigm, which are: volume (ever increasing amount of data to be processed), velocity (data generation at fast and unpredictable rates), value (huge value but hidden in massive datasets at very low density), and volatility (the acquired data streams must be transported and processed in real time). They described the volatility as 5^{th} V. The volatility is used in featuring big data stream applications. According to these 5 Vs characterization,

the life cycle of big data streams can be composed of three phases: data acquisition and local preprocessing at the mobile devices, data transport, and data post-processing at the remote data centers [50]. To gain user trust, a tight interpretation is required. To create a valuable knowledge from a large qualitative data, data mining techniques must be intervened [45]. The main challenge is the extraction of useful information from a complex sensing environment [1]. For that reason, advanced data mining tools are needed. Data needs to be understood using computational and mathematical models [4]. The shallow learning methods are those that are most currently used. They attempt to make sense of data by using supervised and unsupervised learning for extracting pre-defined events and data anomalies [1]. The major research problem in complex sensing environments is how to simultaneously learn representations of events and activities at multiple levels of complexity. Deep learning, the branch of machine learning is based on algorithms that attempt to model multiple layers of abstraction that can be used to interpret given data. Deep learning has to construct adaptive algorithms, distributed, and incremental learning techniques responding to resource constraints in sensor networks [1]. To realize a reliable interpretation and modeling knowledge to get good comprehensible and usable knowledge, many challenges must be treated. When addressing noise, a reliable signal processing technique must be applied to obtain a pure signal. Developing new and efficient inference techniques must be realized. As some of the same data streams can be used for many different purposes, data issuers and how the data was processed must be known [45].

1.5.9 Storage

The storage deficit of the IoT devices results in the emergence of the CloudIoT paradigm. Cloud computing solved this problem by constantly receiving a huge stream of data. In large scale networks, some network problems (like burstiness) may occur as well as the processing load problem. For that reason, data has to be timestamped, so servers can process data by avoiding the problems of processing the load and the burstiness problem. The predictive storage and caching can also address such issues [41]. Constantly sending a huge number of data from sensors to the Cloud for storage provokes the unbalanced usage of resources problem. Over time, some servers will be extremely overloaded while others remain inactive. This imbalance may negatively

affect the response time of the data input and output. To solve such a problem, load balancing techniques must be applied to respond better and faster to client requests. Load balancing techniques distribute the workload evenly across multiple computers in such a way that no processes are overloaded but they are sharing the load. If some servers become overloaded, it establishes links between the underloaded servers and the overloaded ones to redistribute their load. Load balancing aims to increase the system performance by maximizing the throughput, minimizing response time, avoiding overload, achieving the optimal resource utilization, and providing high user satisfaction [51–53].

1.5.9.1 Types and characteristics of load balancing algorithms

— **Static and dynamic load balancing**

Static load Balancing: In the task assignment process, this approach does not take into consideration the current state of the system, but its previous knowledge [6, 54–56]. The static algorithms should be applied in a homogeneous environment where the resources should not be flexible. They are not suited for a heterogeneous environment. Changes in such environments are not accepted during run-time. The tasks are served as First In, First Out (FIFO) [54].

Dynamic load Balancing: Unlike the static approach, the dynamic load balancing decisions depend on the present state of the system where the previous knowledge is not required [6, 54–56]. This approach supports heterogeneous and flexible resources [54]. It is also scalable and fault tolerant. On the other hand, it is complex and time consuming [57].

— **Stability**

According to [58], stability is the time gained when applying load balancing by obtaining faster performance by a specified amount of time. It can be also characterized by the delays in the information transfer between processors.

— **Waiting time**

The waiting time in load balancing should be always less. It is interpreted by the time period elapsed in waiting in the ready queue [59].

— **Cooperative**

Cooperative makes all the processors responsible to execute their own portions of the scheduling task. They also share information between themselves in making the process allocation decision. In other words, they all work together to achieve a common goal, which is efficiency [58].

— **Process migration**

The algorithm with an enabled progress migration parameter is able to decide when it should make changes to the load distribution during execution of the process. It is also able to decide whether to create the process locally or remotely. It is aware when a system will transfer its process when necessary [58, 59].

— **Centralized or decentralized**

It describes the structure of the algorithm (centralized or decentralized) [59].

— **Resource utilization**

This parameter indicates whether the algorithm is able to utilize all the resources in an optimized manner. In other words, all processors share the workload [58,59].

— **Fault tolerance**

This parameter enables the algorithm to keep working properly in the event of some failure. An algorithm with fault tolerance is also able to tolerate tortuous faults [58].

1.5.9.2 Algorithms for load balancing

1.5.9.2.1 Round Robin algorithm: In the Round Robin algorithm [52, 58, 60] the workload is distributed evenly across all processors. Each new process is assigned to a new processor in round robin order (as shown in Fig. 1.9).

The advantage: The Round Robin algorithm does not require inter-process communication. It is very useful for jobs of equal processing time and servers with the same capacities [59].

The drawback: This algorithm does not take into consideration the job processing time. It also assigns the jobs without considering the current load on each virtual machine [61].

Since the processing time of any job is not known in advance, even though the workload is distributed evenly among the servers, this does not mean that the load is tightly balanced. At any point of time some servers may be overloaded while others remain inactive. This is due to the non-equivalence of the job processing time. This problem increases the response time and minimizes the resource utilization which degrades the system performance [6, 52, 58, 60, 61].

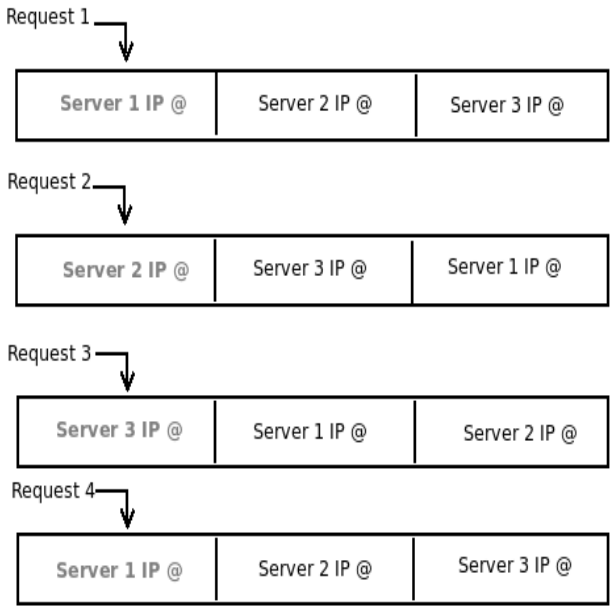

Figure 1.9: Process assignment in round robin order.

1.5.9.2.2 Random algorithm: According to [59, 62], when an overloaded site wants to export its surplus load, the destination will be selected randomly without exchanging the load information. In other words, no policy is used for process migration (as indicated in Fig. 1.10). The shortcoming of this technique is that it guarantees neither the system stability nor better utilization of the system's resources. It also causes unnecessary communication overheads.

1.5.9.2.3 Central Manager algorithm: According to [58,59], a central processor is responsible for the task assignment into different hosts. Once the process is created, the processor which has the minimum overall load will be selected for this new process. Information on the overall load is collected by remote processors by periodically sending their new load state when their load changes (See Fig. 1.11).

1. The client sends a request to the load balancer.

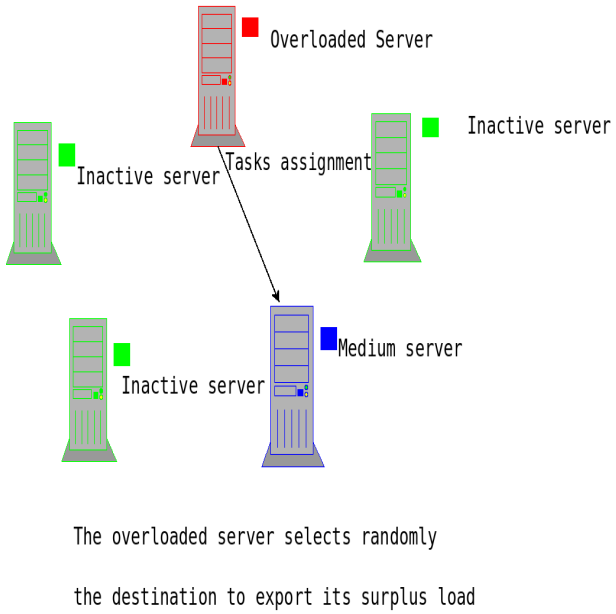

Figure 1.10: Random algorithm.

2. The load balancer searches and selects the ideal server.
3. Task assignment.

1.5.9.2.4 Load Vector algorithm: According to [52, 62], each site maintains a vector that contains information about its local load and the minimum load of the other sites. Periodically, this vector will be sent to another site which is randomly selected to upload the system's load information, taking into account the information stored by the vector.

The advantage: Minimized communication overhead [59].

The drawback: Algorithm is not scalable [59].

1.5.9.2.5 Central Queue algorithm: The Central Queue algorithm [58–60] handles the unfulfilled requests and the activities by using a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is entered in the queue. Then, whenever a request for an activity is received by the queue manager, it removes the first activity from the queue and sends it to the requester. So, the oldest requests are prioritized to be handled before the newest one. In case of the absence of the ready

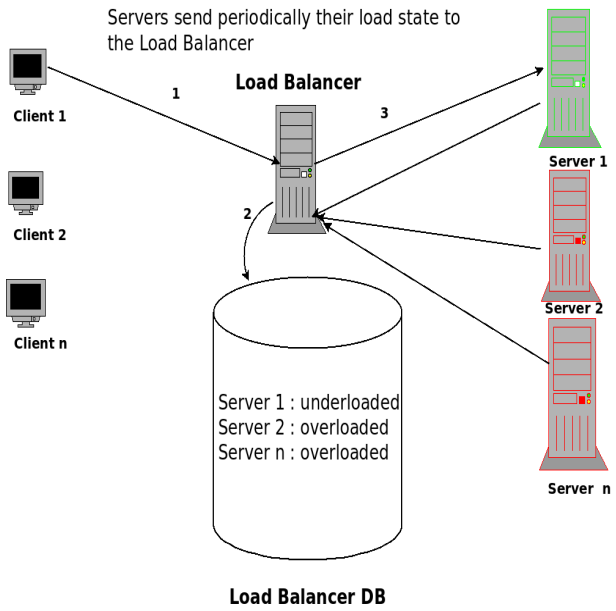

Figure 1.11: Central Manager algorithm.

activities, the arriving requests will be buffered in the queue until a new activity is available. In such cases, if a new activity arrives at the queue manager, the request is removed from the queue and the new activity is assigned to it. When a processor workload overtakes the required threshold, the local load manager sends a request for a new activity to the central load manager. The latter replies to the requests if a ready activity is available in the process-request queue. In the opposite case, it queues the request until a new activity becomes available.

The advantage: the central queue algorithm is useful and adequate for heterogeneous nodes [59].

The drawbacks: the central node algorithm suffers from the single point of failure. It does not deal with the inter-process task migration [59].

1.5.9.2.6 Threshold algorithm: According to [58, 59], this algorithm, initially assumes that all processors are in an underloaded state. The hosts for the new process' assignment are selected locally without transmitting remote messages. If the local load state of this host is overloaded, this host transfers the process to a heavy load host in the system. When the overloaded host does not find a heavy node in the system, the process will be allocated locally. All the information about the system's load are known

in each site by recording a private copy of the load state of each machine. When a site becomes overloaded, it sends a message to all machines to make them aware of its new load state. The processor load state can be underloaded, medium, or overloaded. The Th_upper and Th_under are two predefined thresholds that identify the processor's workload in which:

Under loaded: $load < Th\_under$

Medium: $Th\_under \leq load \leq Th\_upper$

Overloaded: $load > Th\_upper$

The advantages: It has low inter-process communication. This algorithm achieves high performance because of the huge number of local process allocations which results in low inter-process communication. Hence, it decreases the overhead of remote process allocations and remote memory accesses [58, 59].

The drawbacks: The overloaded local host handles the process when all the system's hosts are overloaded. This may provoke two main problems [58, 59]:

1. The overloaded hosts may not have the same surplus load; some servers are extremely loaded more than others. This causes a considerable disturbance in the system.
2. It remarkably increases the execution time.

Devi [63], proposed an architecture which is defined between the Cloud servers and the client requests. It performs the allocation of the processes to different Clouds in underload and overload conditions.

DNS-based approach [53, 64]. This algorithm employs a single virtual interface called a cluster DNS, which is the only one responsible for spreading and delegating the requests among the servers.

Load balancing algorithm based on ***Round Robin in Virtual Machine (VM)*** environment [65], this algorithm aims to achieve better response time and processing time by using a technique based on fuzzy logic.

The authors Desai and Prajapati of the paper entitled, 'Central Load Balancing Decision Model' [66] implemented the Genetic Algorithm to realize the load balancing

in the Cloud-based multimedia system.

Duel Direction Downloading Algorithm from FTP servers [66]. This algorithm presents a dynamic load balancing technique named Duel Direction Downloading Algorithm from FTP (DDFTP) server. In DDFTP, a file of size Z is divided into Z/2 partitions and each node processes the task. The aim of this protocol is to reduce the overhead.

In [66] the authors proposed a Honey Bee behavior inspired Load Balancing [HBB-LB] technique. The inspired technique helps to achieve even load balancing across virtual machines to maximize throughput.

1.5.9.3 Comparison of algorithms

From Table 1.2 we remark that all the static algorithms are stable. We conclude that the stability is correlated with the static state of the algorithm.

Table 1.2: Comparative analysis of load balancing algorithms

	Round-Robin	**Load Vector**	**Random**	**Threshold**	**Central-Algorithm**	**Central-queue**
Stability	Stable	Stable	Stable	Stable	Stable	Small
Fault tolerance	No	No	No	No	Yes	Yes
Resource utilization	Less	Less	Less	Less	Less	Less
Cooperative	No	No	No	Yes	Yes	Yes
Process migration	Not possible	Not possibl	Not possible	Not possible	Not possible	Not possible
Centralized/ Decentralized	D	D	D	D	C	C
Dynamic/Static	S	S	S	S	S	D
Waiting time	More	More	More	More	More	Less

1.5.10 Availability

Webster's dictionary defines availability as 'the quality of being present or ready for immediate use' [67]. Making the services available at any moment and anywhere for customers requires the achievement of the availability of hardware and software. Available software is software that is able to deliver services for everyone at different places simultaneously. The hardware availability is the existence of the compatible

devices with the IoT functionalities and protocols [25].

The main challenge in designing extremely available systems is to tolerate each failure as it occurs and recovers from its effects. Making informed decisions about the availability of each individual system component is required [67]. Because of the IoT devices are prone to failure [1], fault tolerance techniques and redundancy for devices and services are required. When using redundancy, the hardware availability can ensure the availability of the application. For example, when a component is out of order, the service can be achieved by another component. The problem that arises when using redundancy is the cost generated by the components. Furthermore, there are some studies about the availability evaluation of the IoT applications. These studies help in boosting the availability of the IoT system [2, 25].

1.5.11 Reliability

The reliability describes the ability of a system to work correctly according to its specifications. Reliability and availability are not independent. In other words, the availability of services and information can be achieved over time by the insurance of the reliability [25]. In critical applications, the communication network is considered as an essential and critical part that must be fault-tolerant. This property is the key enabler for getting a reliable distributed information [25]. When the communication in the IoT system is not reliable enough, some problems can occur in data gathering, data processing, and the real-time transmission. These problems can lead to a wrong decision which can result into disastrous scenarios in the system [25]. Reliability can be achieved at:

— the data transmission level by minimizing the packet loss rate and minimizing the packet latency (especially in emergency applications),

— the data aggregation level by reducing the network congestion,

— the sensing level by choosing a reliable capturing.

In 2013, Maalel *et al.* [68] proposed a reliable routing protocol with a reliable data transmission in the IoT emergency applications.

In 2016, Yi and Yang [69] proposed a Hamilton Energy-Efficient Routing Protocol (HEER). It is an improved delay-aware and energy-efficient clustered protocol. The

HEER forms its clusters in the network initialization phase. Links that belong to each cluster on a Hamilton path are established using a greedy algorithm.

For facilitating the evaluation of discovering routes, Shao *et al.* in [70] presented a novel routing metric, called Coding-aware QoS Routing Metric (CQRM). The CQRM jointly considers node congestion, link quality, and coding opportunity.

In 2015, Li *et al.* [71] applied an improved Nonlinear Dynamic Adaptive Particle Swarm Optimization (NDAPSO) algorithm for producing energy-aware clusters with a selection of optimal cluster heads.

1.5.12 Management

The connection of a large number of devices in the network makes the management of the Fault, Configuration, Accounting, Performance and, Security (FCAPS) aspects of these devices difficult. To treat such a problem, lightweight management protocols must intervene. This management is a key solution that contributes to the growth of the IoT deployments [25]. The persistent reliability of the Machine to Machine (M2M) communication is required for customers' satisfaction. The Open Mobile Alliance aims to provide M2M applications with remote management capabilities of machine-to-machine devices, services, and applications [25, 72]. It developed the Lightweight M2M (LWM2M). It is a device management protocol which implements an interface between the M2M device and the M2M server. This interface is intended for the management of a diversity of devices [25]. The Network Configuration Protocol (NETCONF) is defined by the IETF to 'install, manipulate, and delete the configuration of network devices'. The NETCONF operations are realized on top of a Remote Procedure Call (RPC) layer using XML encoding and provides a basic set of operations to edit and query the configuration on a network device [25, 73].

1.5.13 Interoperability

Dealing with many heterogeneous devices deployed in the IoT that belong to different platforms results in the emergence of some interoperability challenges [25]. Because the IoT elements lack storage capacity, the captured data must be transmitted from sensors to the Cloud. The problem occurs when Cloud computing and the IoT objects implement non-standard heterogeneous interfaces [41]. To gain custom- ers' satisfac-

tion, all must benefit from the IoT's services whatever the platform they use. For that reason, interoperability should be considered by both application developers and the IoT's device manufacturers to guarantee the delivery of services for all customers. Programmers of the IoT should build their applications to allow for adding new functions without causing problems or losing functions while maintaining integration with different communication technologies. It is noted that interoperability is a significant criterion in designing and building the IoT services to meet customers' requirements. To treat such a problem, the Center for Testing and Interoperability (CTI) offers a wide range of services for testing and interoperability [25].

The European Telecommunications Standards Institute (ETSI) Plugtests test the interoperability between different implementations which enables the assessment of the interoperability level of the tested products. These solutions support the development of new technologies [25, 74].

PROBE-IT: This is a European research project that aims at ensuring the interoperability of validated IoT solutions [25, 75].

1.5.14 Robustness

A robust system must have the ability of tolerate perturbations that might affect it. So, any deployed system must have integrated techniques to deal with failure. Because the IoT applications are entirely based on sensors, actuators and the communication network, all these devices must know their locations, and must have a synchronized clock. Also, each device must be aware of the set of devices that it has to communicate with. Over time, these conditions can be deteriorated. Formal methods to develop reliable code, on-line fault tolerance, and in-field-maintenance are the principal methods that aid to achieve a consistent service. The principal reasons that contribute to creating the cited problems are: the unexpected operation of the system, the longevity, the openness of the systems, and the realities of the physical world [45]. A robust system must be built in spite of noisy, and faulty, and unexpected physical world actualities [45].

1.6 Conclusion

The large scale networks that embedded a number of fully connected IoT devices enabled the emergence of new applications in diverse domains. The IoT aims to convert our ordinary world into a smart one by facilitating our life style. The creation of a reliable and performance smart world will be met by some challenges that negatively affect the system performance. Architecture, privacy, security, openness, standardization, new protocols, new standards, big data, and creating knowledge, mobility and scalability are the main challenges that must be solved. Energy also must be tightly conserved to extend the longevity of the IoT network. To successfully achieve an efficient critical system, the quality of services must be taken into account. To confront these challenges, relevant studies must be done. Hence, for a comprehensible understanding, this chapter first presented the IoT's architecture, its basic elements and its research trends. Then, it summarized these challenges raised in recent literature. It also presented appropriately proposed solutions and the important research directions. The quality of service criteria and the proposed solutions are also discussed.

Table 1.3 illustrates the research that provides the common problems and the essential requirements that should be realized while designing the IoT architecture.

The next chapter overviews the related work that achieves the quality of services of IoT systems. It also presents the related work and techniques that treat the commun challenges that the IoT elements suffer from.

Table 1.3: IoT main research challenges

Challenges	***References***
Architecture	[1, 4, 20, 27, 45]
security	[1, 4, 20, 25, 41, 45, 46]
Openness	[45]
Privacy	[1, 4, 20, 25, 41, 45, 46]
Standardization	[20, 34, 41]
New Protocols	[1, 32, 33, 41]
Energy	[1, 41, 48]
Big data and creating knowledge	[1, 41, 48]
Availability	[2, 25, 67]
Reliability	[25, 41, 68–71]
Management and performance	[25, 41]
Mobility and scalability	[25]
Visualization	[38–40]
Storage	[6, 7, 41, 52, 54–61, 63] [64–66]

Chapter 2

Problem Statement and Related Work

2.1 Introduction

We discussed in chapter one that robustness, availability, reliability and management of faults and a performance system are the main quality of services that contribute to the growth of IoT deployment. Wireless sensors and Cloud computing are the main elements of Internet of Things. Cloud servers are dynamic, which makes them prone to failures. Managements of faults can be achieved by applying the appropriate fault tolerance techniques. Reliability can also be achieved at the data transmission level by minimizing the packet loss rate and minimizing the packet latency (especially in emergency applications), and at the data aggregation level by reducing the network congestion [25]. Availability is making the services available at any moment and anywhere for customers. The main challenge in designing extremely available systems is to tolerate each failure as it occurs and recovers from its effects. Making informed decisions about the availability of each individual system component is required [67]. Because of the IoT devices are prone to failure [1], fault tolerance techniques and redundancy for devices and services are required. When using redundancy, the hardware availability can ensure the availability of the application [2] [25]. In addition, energy and storage are the most common challenges that affect the IoT elements. The IEEE 802.15.4e creates open communication standards of the IoT. It creates four modes dedicated to IoT elements and applications. Time Slotted Channel Hopping (TSCH) is the

appropriate mode designed for Wireless sensors constraints. It provides deterministic latency to applications, and increases the network's capacity. It introduces the mechanism of multi channel hopping that reduces the external interference and multi-path fading problems [68]. On one hand, it hops from channel to another in indiscriminate way, which risks the transmission to be failed because of the external interference. The TSCH uses shared links to allow more than two communicating nodes to transmit packets at the same time. Thus, it increases the network throughput. On the other hand, the shared link nature may cause internal collision problems that seriously affects sensors' energy.

Beside the energy deficit and the power transmission of sensors, the storage of data capture by sensors is also considered a big challenge. Despite the availability of the massive storage capacity that Cloud computing servers offer to store big data captured by sensors, the random or the unstructured stored data opens another challenge which is the increase of requests latency time. The dynamic environment of Cloud results in various unexpected faults and failures, thus achieving a robust system is also challenging.

This chapter addresses the IoT systems based on sensors and Cloud computing challenges. It presents the related work, and techniques that address the increase of latency time by avoiding data retransmissions and providing structured data storage. Also, it discusses the related work about Wireless Sensors Networks reliability. Chapter two is divided into two parts. The first part treats latency, data availability and fault tolerance techniques, at the level of Cloud computing. It also explains the importance of Multi-agents systems in complex systems by describing their roles. The second part gives an overview about TSCH, presents a related work on how to achieve reliable systems and overcome external interference and multi-path fading in the environments that have co-existence of other networks .

2.2 Load Balancing

This part discusses some algorithms and the related work that address the load balancing and Cloud computing reliability.

2.2.1 Load Balancing Approaches

Centralized load balancing and distributed load balancing are the common approaches used to achieve an optimum load balancing among servers.

— **Centralized load balancing:** A single load balancing server gathers workload information for each server. By assembling the information set of each server, the single server can determine the system's global state [76].

Advantages: Controls carefully the system by assuring the ideal load balancing, and providing fast requests' responses.

Drawbacks: This approach has a single point of failure caused by the central server.

— **Distributed load balancing:** Each server in the system is responsible for collecting workload of other servers and all registered information to obtain the global state [77].

Advantage: It solves the single point failure problem.

Drawback: It generates large number of overhead messages which affects the latency negatively.

2.2.2 Load Balancing Algorithms

— **Min-Min Load Balancing Algorithm:** In this algorithm the jobs with least execution time are assigned or processed first while the other jobs (with the longest execution time) are processed last [78]. The jobs are assigned to the host that are capable to complete all the tasks in a minimum time [57]. The two main drawbacks of this algorithm is that before the task assignment the Cloud manager does not check the selected host to see whether it is in a busy or an idle state. Also, over time this algorithm may lead to the starvation problem [78].

— **Opportunistic Load Balancing Algorithm:** This algorithm tries to keep every node busy. It assigns the unassigned tasks in random order to the free nodes. Because this algorithm does not calculate the existing execution time of the node, the algorithm may not achieve better results in term of the speed of the execution time [78].

— **Active Clustering Algorithm:** The idea behind this algorithm is that the similar nodes are grouped together to work in these groups. The algorithm is based on some steps that should be iteratively executed [57].

2.2.3 Load Balancing related work

K. Nishant et al [66] proposed a static load balancing technique called Ant Colony Optimization. This technique is based on the Ant behavior to collect information of Cloud nodes.

Pradeep Naik, Surbhi Agrawal, and Srikanta Murthy in [79] proposed a new model that employs a bipartite algorithm graph. The task assignment to a node will be established by matching the request and resource information, using a maximum matching bipartite algorithm.

The authors in [80] proposed Trust Assisted Sensor Cloud (TASC) which consists of trust-assisted WSN and trust-assisted Cloud. It utilizes trust sensors to gather and send sensory data to the Cloud. Only the sensors with trust values exceeding a threshold are allowed to transmit their data to the Cloud. The Cloud computing system selects the trusted data centers to store and process the sensed data and further transmit data to end users on demand. The trusted data centers are those with trust values exceeding a threshold. The trust-assisted WSN and the trust-assisted Cloud values are calculated based on using respectively the historical trust of sensors and the historical trust of Clouds. Taking into account the historical trust when estimating the sensor current trust and the Cloud current trust gives more accuracy to the application. The shortcoming of this strategy is that a zone risks to be uncontrolled when all its nodes are distrusted for a long period.

The authors in [63] proposed an architecture which allocates the processes to different Clouds. A Cloud is assigned according to its under-load and overload conditions. The overload conditions are treated using the concept of process migration from one Cloud to another. Establishing these conditions ensures load balancing among servers. It also realizes a high service availability. On the other hand, migrating the processes from a Cloud to another needs a high security attention.

2.2.4 Fault tolerance using Replication

Redundancy is considered as one of the main solutions to achieve fault tolerance against crash faults, thus provides scalability, reliability and availability of the system performance [53]. In redundancy technique we distinguish two kinds of scenarios: Active-Active Scenario and Active-Standby Scenario [53].

Active-Standby Scenario: In this case we have two servers: the primary server and the backup (replicas) server. Only one server processes the requests while the other one monitors and saves the system state. If the primary server fails, the backup replicas determines its failure and takes its place [53].

Active-Active Scenario: Both primary and replicas servers accept traffic. All the replicas servers are simultaneously invoked and each replica processes the same request at the same time, in such a case, when one of these servers fails the system can continue to serve its services [53].

2.2.5 Time Triggered Protocol (TTP)

Time-Triggered Architecture (TTA) is an architecture dedicated to distributed computer for high-dependability real-time systems. The keystone of the TTA is the Time-Triggered Communication Protocol TTC/P. The TTC/P provides message transmissions, clock synchronization and Group Membership Protocol (GMP) fault tolerant services to ensure safety-critical, fault-tolerant high-speed networks [81,82]. In TTC/P each computer is assigned to one slot during which it can broadcast its message (i.e. in Time Division Multiple Access) [81].

2.2.6 Event Triggered Protocol (ETP)

Unlike the event triggered system where its activities are initiated periodically at predetermined points in real-time, the event triggered protocol activities are initiated as a consequence of the occurrence of a significant event [83].In ETP, each node uses its local information to determine when to make a transmission [84].

2.2.7 Multi-Agent Systems

"Systems composed of multiple interacting computing elements, known as agents. Agents are computer systems with two important capabilities. First, they are (at least to some extend) capable of deciding autonomously for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day life: cooperation, coordination, negotiation, and the like " [85].

According to [86] software intelligent agents are classified into three categories: Information agent, cooperation agent and Transaction agent.

— **Information agents:** they provide a useful assistance for their clients in the search for a required information in networks or distributed systems [86].

— **Cooperation agents:** They solve hard and complex problems; when an agent is not capable of processing a task alone, or agents already have the solution of a problem. Cooperation agents achieve their tasks by employing cooperation mechanisms and communication with other objects [86].

— **Transaction agents:** They operate in sensitive areas (e.g., classical database environment, network management) that require a high degree of responsibility. They perform the monitoring and the processing of transactions to ensure trustworthiness, security and data protection robustness [86].

2.3 Time Slotted Channel Hopping (TSCH)

This section overviews the TSCH protocol. It defines the TSCH, the time slot, slotframe, node TSCH scheduling and channel hopping.

2.3.1 TSCH Definition

The TSCH mode aims to achieve an efficient exploitation of the available network resources. TSCH process is designed to be implemented in a critical application such as: climate control, oil and gas industry applications, food and beverage products, chemical products, pharmaceutical products, water/waste water treatments, and green energy production. This mode combines the time slotted access with multi-channel capabilities to provide deterministic latency for applications and to achieve a greater throughput capacity with collision-free communication. These benefits are achieved by allowing sensors to communicate simultaneously, but using different frequencies. Moreover, it fully supports low-power operation as it always maintains the time synchronization. This mode has the potential to be beneficial in routing (transmitting) the data in a reliable way. It consolidates the network topology by stabilizing all the links of the network. The latter efficiently protects the links from the common wireless phenomena like the external interference, and the multi-path fading that highly affects the applications deployed in the harsh environment (e.g., industrial application) [14, 45, 68]. TSCH is topology independent. The network topology can be a star topology, tree topology, partial or mesh topology [14, 68].

2.3.2 Time slot, Slotframe, node TSCH Scheduling

2.3.2.1 The Time slot

The time slot is a defined interval of time designed for specific operations like: the sending of a maximum-size data frame from the sender to the receiver, the reception of the associated ACK from the receiver to confirm the successful reception of data, packet processing, security operation, and the radio turnaround [14].

2.3.2.2 Slotframe

A slotframe is a block containing a number of time slots that automatically and cyclically repeat over time. Typically, the slotframes and a device's assigned time slot(s) within the slotframe are configured by a higher layer at the time when the device joins the network. Each node knows the necessary information to communicate via the enhanced beacon (EB) frames. The EB frames include synchronization, channel hopping, time slot and slotframe information. The EBs are sent periodically by other nodes for the network advertisement. When the nodes receive the EB, they synchronize with the network and start sending their own beacons. Once all the nodes in the network are synchronized, they will be able to communicate with no need to use the enhanced beacons instructs(i.e. the slotframe will be automatically repeated based on the participating devices' shared notion of time) [45, 68]. In the TSCH mode, the slotframe size varies from 10 to 1000 time slot and its size depends on the application needs [14, 68]. There is a strong correlation between the slotframe size, the availability of the bandwidth resources, and the power consumption. When the slotframe size is short, the time slot repeats more often, and more bandwidth resource is available, but the energy consumption gets significantly high [14] (Figure 2.1).
ASN is the number of time slot since the start of the network.
TS_i is the time slot number within a slotframe.

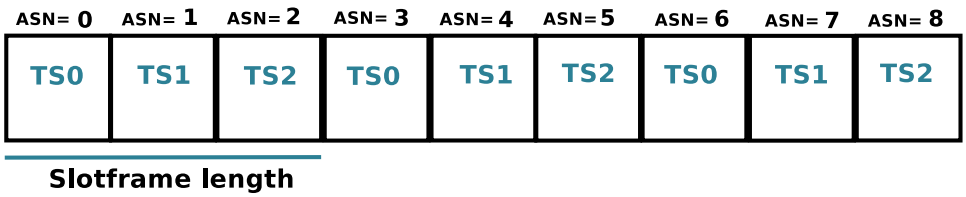

Figure 2.1: TSCH slotframe structure

2.3.2.3 TSCH Node Scheduling

Typically the TSCH schedule is considered as a mathematical matrix. The matrix rows' number is equal to the slotframe size (the number of the time slots) and the matrix columns' number is equal to the number of frequencies (the number of the channelOffset). The container of this matrix is a set of cells. TSCH schedule assigns different activities to nodes in each time slot. During a time slot a node can play one

of three roles/activities (transmit, receive, or sleep). Each cell offers a "quantum" of bandwidth to the communicating nodes. A cell of a node that is chosen to transmit or to receive data in a time slot is defined by a ChannelOffset on which the sender and the receiver communicate. For each transmission cell, the node verifies its outgoing buffer to check whether there is a packet to deliver or not. If yes, this node transmits the packet to the neighbour's address written in the schedule information for that time slot and waits the reception of the acknowledgement. If there is no message in the node's buffer, the node turns its radio off for the duration of the time slot in order to save energy. For each reception cell, over a predefined period the node listens to the channel to receive the incoming packets. If the node receives a message during this period, it checks it and replies with an acknowledgement message; otherwise, it switches its radio off [14, 45, 68].

2.3.3 Channel hopping

One of the main efficient techniques that the TSCH applies is the channel hopping mechanism. It allows devices to hop over the channel space to mitigate the negative consequences of multi-path fading and interference. It also, allows several simultaneous communications to occur as long as they use different channel offsets. Typically, the TSCH mode uses 16 channels for communication. To identify the channels, each channel is assigned a unique integer value between 0 and 15. This value is called channelOffset. The channels with poor communication quality is not used. Thus, the number of usable channels (Nchannels) may be less than 16. A link between two communicating nodes is identified by a slotOffset and a channelOffset. When a node A is selected to transmit data to node B in a predetermined slotOffset and channelOffset, the node B will systematically receive the data at the same slotOffset and in the same channelOffset. The frequency (physical channel) in a shared or a dedicated link is calculated according to equation (1) [14, 45, 68].

$$f = F[(ASN + channelOffset)\%Nchannels] \quad (1)$$

NChannels: is the number of the available channels (size of the lookup table F)

ChannelOffset: is the channel number.

ASN: is used in the determination of the total number of the time slots elapsed since the start of the network. This number is globally incremented at every time slot.

F: is a lookup table containing the set of available channels.

Figure 2.2 shows the possible cells scheduling for data collection in a simple network with a tree topology.

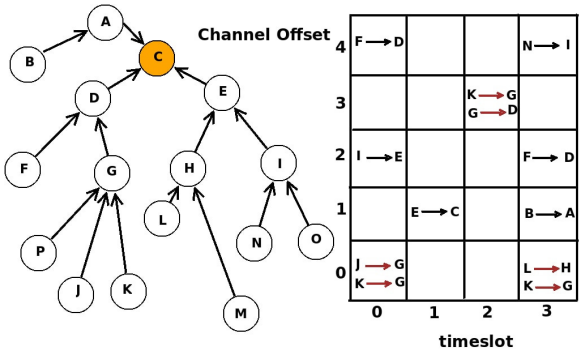

Figure 2.2: Cells scheduling for data collection with a tree network topology

We suppose that there are only 5 available channels for communication and the 11 remaining channels are blacklisted because of their poor communication quality. There are two types of links: dedicated links and shared links. The former is allocated to only one node for a transmission. The latter is intentionally allocated to more than one node for a transmission (links in cells [0,0], [0,3], and [3,2]).

2.3.4 Shared links

On one shared link multiple nodes can transmit data at the same time slot and in the same ChannelOffset regardless of their physical positions. When a link is shared by two or more interfering nodes, the packets will collide with one another when being transmitted. The transmission failure is detected by non receiving the expected acknowledgement.

Example: In the case [0,0] , when node J sends to node G and node K sends to node G using the same link, an inevitable collision will happen at the destination G. In this case, the failed packets will be retransmitted. The IEEE 802.15.4e standard proposes a TSCH CSMA-CA algorithm that reduces the probability of repeated collisions.

2.4 TSCH CSMA-CA retransmission algorithm

The IEEE 802.15.4e standard proposes a CSMA-CA retransmission algorithm (**Algorithm 1**) that reduces the probability of repeated collisions when the packets are retransmitted. In a shared link, when a transmission from K to G fails, the node K implements a backoff algorithm and waits for another link to the destination device. The subsequent retransmission may be on either shared links or on the next dedicated link to the destination device.

This backoff algorithm has the following parameters:

NB is the number of consecutive failed retransmissions on a shared link. It is incremented in each retransmission attempt for the on-going frame.

MacMaxFrameRetries is the maximum allowed number of retransmissions of a packet.

BE is the backoff exponents. It is initialized to *macMinBE*. The *macMinBE* value is in the range of 0 to *macMaxBE*. The *macMaxBE* is defined by the standard. It is an integer value in the range of 3 to 8.

W is a randomly generated number of a shared transmission link. This number is selected randomly in the range of 0 to $(2^{BE} - 1)$.

2.5 Priority Channel Access Backoff Algorithm

A device in a TSCH PAN (Personal Area Network) may also use the Priority Channel Access Backoff Algorithm (PCA) method for critical event messages. When a critical event packet fails to be transmitted, the standard defines a PCA backoff critical event packet retransmission algorithm. When a critical event packet is not successfully transmitted, the backoff algorithm postpones the retransmission for TB+1 shared links to destination R, or to the nearest dedicated link located before (less than) the TB+1 shared links to destination R (**Figure 2.3**)).

BE is initialized to the value of $macMinBe-1$. BE remains constant for subsequent retransmissions.

$macMinBe$ = [0, $macMaxBe$]. The default value is 1.

$macMaxBe$ = [3,8]. The default value is 7.

TB is a variable maintained by the MAC sub-layer. It indicates the number of

Algorithm 1 The TSCH CSMA-CA algorithm

BEGIN
Variables
$NB := 0$;
BE:= *macMinBE*;
W: Interger;
Generate a random number W in $[0, 2^{BE} - 1]$
Delay for W shared links
Retransmission data in W shared link
while the retransmission is not acknowledged **do**
  NB:= NB+1;
  BE:= Min (BE+1, *macMaxBE*);
  if $NB >$ *macMaxFrameRetries* **then**
    The Drop of the packet.
    The end of the algorithm.
  else
    Generate a random number W in $[0, 2^{BE} - 1]$.
    Delay for W shared links.
    Retransmission data in W shared link.
  end if
end while
if The retransmission is acknowledged **then**
  BE:= *macMinBE*;
end if
END

remaining backoff periods since the start of the CSMA-CA with PCA backoff algorithm. The variable is initialized to a random value between 0 and (2^{BE}-1). It is used in order to gain access to the channel in a timely manner.

MacCritMsgDelayTol : an integer attribute. It indicates the maximum transaction delay in milliseconds for a critical event message. The $MacCritMsgDelayTol$ is in the range of [0,65532]. The default value is 15000.

macMaxFrameRetries: The maximum number of attempts allowed after a transmission failure. This attribute is in the range of [0,7]. The default value is 3.

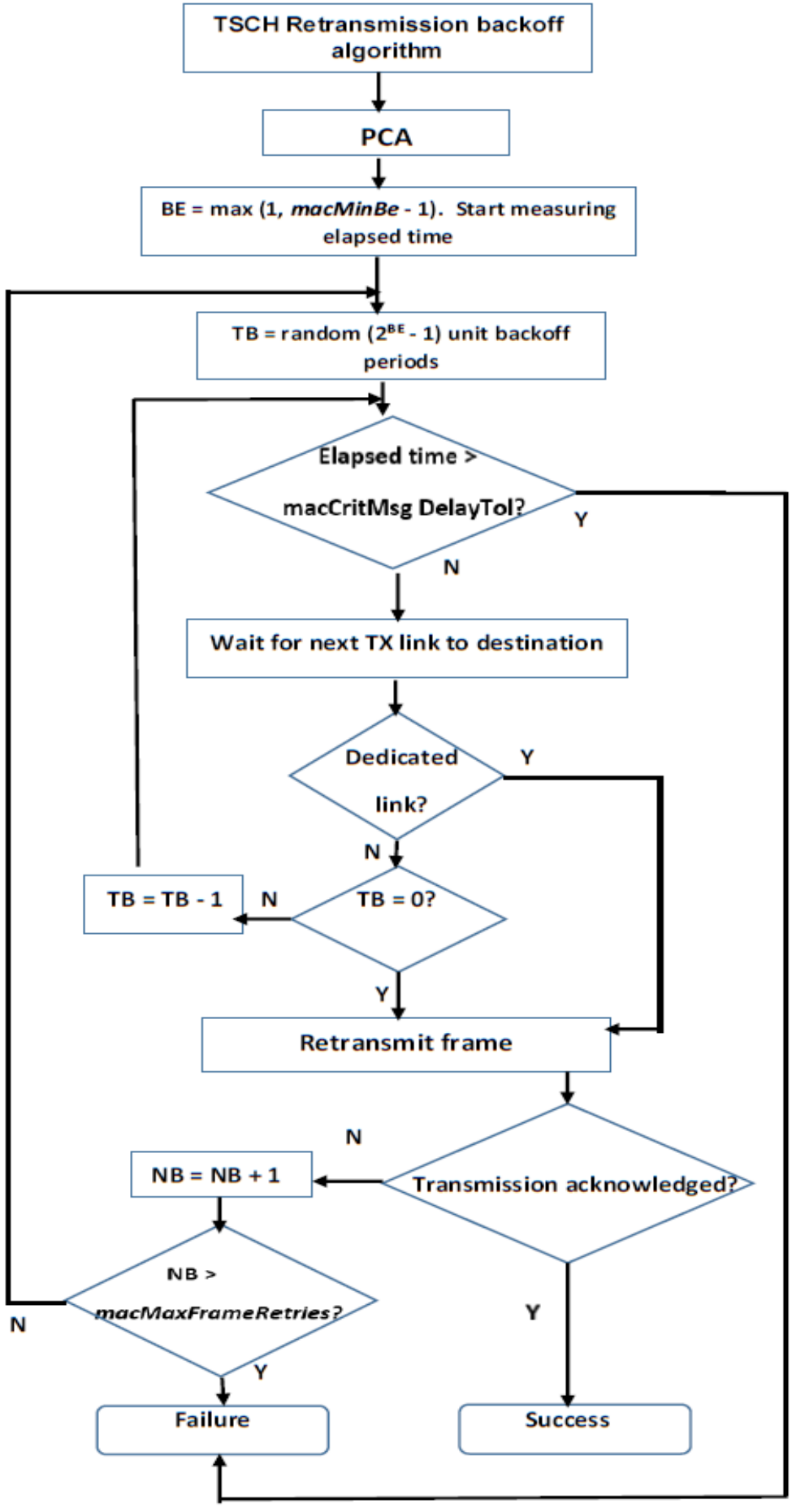

Figure 2.3: Organigramme of the PCA backoff retransmission algorithm

2.6 Networks' reliability related work

Reliability is the ability of the WSN to guarantee that data is always delivered [87]. The main reasons that deteriorate the WSNs reliability are wireless links, route problems and constrained resources like energy [88]. To extend the network's lifetime and to build a reliable WSN, the MAC sub-layer protocol must reduce and avoid as much as possible the idle listening, internal and external interference, and multi-path fading. The MAC sub-layer protocol should tightly mitigate these problems especially in ISM radio bands (Industrial, Scientific and medical radio bands), where the use of ISM equipment generates electromagnetic interference that interrupts radio communications. Space diversity and frequency diversity are common techniques that aim to enforce reliability in WSNs by combating multi-path fading and co-channel interference. The frequency diversity is based on the fact that the signal can be transmitted using several frequency channels. The individual channels experience different levels of interference and fading. Frequency-hopping spread spectrum (FHSS) is a way of exploiting frequency diversity [87]. In the space diversity, the same signal is transmitted over two or several different uncorrelated paths [87].

Some MAC protocols provide node to node reliability while other protocols provide end to end reliability. The node to node reliability is being imposed in small scale networks, between two adjacent nodes. The end to end reliability take place in large scale networks, from the initial source to the final destination [87]. The MAC sub-layer has various techniques and methods to fulfill reliability. The following related works show examples of these techniques.

The authors in [89] adopted a hybrid routing approach that is based on RPL (Routing over Low Power and Lossy Networks) multi-parent trees. The forwarding nodes retransmit only packets received from neighbors with higher rank to avoid loops. The proposed solution is mainly based on FHSS to exploit frequency diversity. It is also based on controlled flooding applied in RPL routing protocol to exploit the space diversity. The FHSS combats external interference and multi-path fading, while the controlled flooding provides for resilience to topology changes and low latency. This protocol enhanced the end to end reliability in WSNs. The proposed algorithm does not take the overhead and the energy consumption problems into account.

The authors in [90] ensured the collision-free reception of data by applying the channel hopping multiple access (CHMA) in a way that no two senders or receivers share the same channel if they are within a two hops neighborhood. The proposed solution avoids collision problems in the presence of hidden terminals. This protocol does not require neither carrier sensing, nor the assignment of unique codes to ensure collision-free data reception. When a node wishes to transmit a local data packet, it sends a ready-to-send (RTS) control packet over the current channel hop to the intended receiver. All nodes hop to the next channel hop. When the receiver successfully receives the data packet, it replies the sender by sending a clear-to-send (CTS) message to proceed the communication. Then, the rest of the nodes hop immediately to the next channel hop. The CHMA has been analyzed for unslotted, multi-channel ad-hoc networks. The CHMA achieved high throughput. This proposed solution boosted the Hop to Hop reliability in WSNs. The major problem of this algorithm is that over time sensors exhibit more energy drain and more overhead when transmitting the CTS and RTS messages.

The authors in [13] mitigated the external interference generated from the co-located wireless systems utilizing the same spectral space by proposing the Adaptive Time Slotted Channel Hopping Technique (A-TSCH) algorithm. They specified a new slot type called the noise floor (NF) intended for collecting the noise floor of each channel. The channel quality is estimated using the exponential smoothing technique (SES). The SES technique employs posterior statistical arguments to carefully evaluate the noise floor of each channel. The A-TSCH algorithm significantly improved the reliability of the channel hopping mechanism and provided for better protection from interference since it avoids using the heavily interfered frequencies. A-TSCH enforces the Hop to Hop reliability in WSNs. The algorithm risks to provide less accuracy of channel estimation since it relies on the RSSI information sensed in just two time slot.

The authors in [43] proposed an enhancement for a medium access control in WSNs based on the overhearing mode. The technique, called Overhearing based MAC (OB-MAC) is used in IEEE 802.15.4 a standard for WSN to minimize the amount of redun-

dant communications. This technique is applied in a dense area where wireless sensors are deployed randomly to obtain low congestion in the network and guarantee shorter communication delays. It supports the fault tolerance since it applies the concept of the influential range. The results showed that the overhearing mode is an efficient method to save energy. It also offers a good latency for WSNs. The OBMAC enforces the end to end reliability in WSNs. For more accuracy in eliminating redundant data transmission and for selecting the most accurate sensed data the influential range must be tightly and carefully selected.

The authors in [88] proposed an algorithm called Reliable Data Collection MAC for WSN (Rainbow) which increases the Hop to Hop reliability in WSN that has tree topology. Rainbow combines the Time Division Multiple Access (TDMA) with the FHSS to achieve tight control channel access. The FHSS is responsible for reducing collision, increasing throughput and preventing interference. The authors presented a light-weight distributed time-frequency channel allocation algorithm, called punch. The latter is used to achieve collision-free communication.

The authors in [91] proposed a multi-agent system which consists of a resource manager, a fault tolerance manager, and load balancing manager. It supports agent mobility over the base station and the sensor node to ensure energy awareness, reliability , scalability, and extensibility. Each node executes a predefined role and shares information with other nodes for WSN management. The proposed algorithm ensures Hop to Hop reliability. The major problem of the algorithm is that the mobile agents which hold critical information are vulnerable to security attacks that may falsify the critical information. This problem leads to system performance deterioration.

Table 2.1 classifies these algorithms according to the techniques employed in MAC protocols to achieve the network's reliability.

Table 2.2 illustrates the type of reliability employed in MAC protocols.

Table 2.1: The Classification of reliable MAC sub-layer protocols

Algorithms	**Frequency diversity**	**Space diversity**	**Overhearing**	**Fault tolerance**
Reliability through TSCH and Flooding-based Routing [89]	√	√		
Channel Hopping Multiple Access (CHMA) [90]	√			
Adaptive time slotted channel hopping (A-TSCH) [13]	√			
Overhearing based MAC [43]			√	√
Reliable Data Collection MAC for WSN (Rainbow) [88]	√			
Multi-Agent system for Fault Tolerance in WSNs [91]			√	√

Table 2.2: Reliability type of MAC sub-layer protocols

Algorithms	Hop to Hop reliability	end to end reliability
Reliability through TSCH and Flooding-based Routing		√
Channel Hopping Multiple Access (CHMA)	√	
Adaptive time slotted channel hopping (A-TSCH)	√	
Overhearing based MAC		√
Reliable Data Collection MAC for WSN (Rainbow)	√	
Multi-Agent system for Fault Tolerance in WSNs		√

2.7 Avoiding external interference related work

This section talks about related works and techniques that exploits the high quality channels and uses the blacklisting techniques.

2.7.1 Link-Quality Estimation Process

link-quality estimation (LQE) is a dominant process that achieves high communication efficiency by mitigating low-power link unreliability. To recognize the unreliability of low-power links, the LQE process demands three steps: link monitoring, link measurements, and metric evaluation [92].

2.7.1.1 Link Monitoring

Link monitoring can be either active or passive. In active monitoring, a node monitors the links to its neighbors by sending probe packets. Probe packets can be sent either in broadcast or unicast mode. Unlike unicast probe packets, broadcast probe packets involve no link-level acknowledgments or retransmissions [92]. The drawback to this method is that probe packets are generally sent at a certain rate, which yields a trade-off between energy efficiency (low rates) and accuracy (high rates) [92]. Unlike active link monitoring, passive link monitoring uses the existing traffic to monitor the link state. Passive monitoring considers the overhead and energy consumption [92].

Several studies [92] state that LQE employing link monitoring based on data traffic is much more accurate than that using link monitoring based on beacon traffic.

2.7.1.2 Link Measurements

Link measurements are carried out by collecting useful information [92] from received packets/acknowledgements or from sent packets. Data extracted from received packets/acknowledgment is used to compute receiver-side LQEs (e.g., sequence numbers, timestamps, received signal strength indicator (RSSI), Link-Quality Indicator (LQI)). Data extracted from sent packets allows the computation of sender-side link-quality estimators (e.g., Sequence numbers, timestamps, and packet-retransmission count) [92].

2.7.1.3 Metric Evaluation

Based on link measurements, a link's quality is estimated using an evaluated metric. This metric is created according to an estimation technique, and can be based on a simple average or a more sophisticated technique such as fuzzy logic, learning, regression, or filtering [92].

The Link Quality Estimation (LQE) is classified into two main classes: hardware-based LQE and software-based LQE. The channel quality Estimation based on hardware is monitored using Received Signal Strength Indication (RSSI), and Link Quality Indicator (LQI) [14] [92].

RSSI can be acquired despite the absence of receiving packets. The LQI can only be acquired during the reception of a packet. The measuring is mainly based on the

analysis of the initial symbols of the received packets [14].

The advantages: because the metric values are provided by the transceiver, the hardware-based estimators do not require computational resources from the sensor nodes [14] [93].

The drawback: relying only on information provided by hardware-based estimators in channels' monitoring is insufficient to provide high accuracy in estimating the channels quality [14] [93].

Software-based estimators allow to approximate either the reception ratio or the average number of packet transmissions/retransmissions before its successful delivery [93]. The most relevant software-based estimators are: the Packet Reception Ratio (PRR), the Required Number of Packet transmissions (RNP), the Window Mean with Exponentially Weighted Moving Average (WMEWMA), and the Expected Transmission Count (ETX) [93].

Tabel 2.3 classifies these software-based estimators according to their location and monitoring type [92].

— PRR metric is the average of the ratio of the number of successfully received packets to the number of transmitted packets. The PRR is computed at the receiver side, for each window of w received packets (see equation 2.1) [93] [92].

$$PRR = \frac{Number\ \ of received\, packets}{Number\ \ of\ sent\, packets} \tag{2.1}$$

— The WMEWMA estimator approximates the PRR estimator (equation 2.2) [92].

$$WMEWMA(\alpha, w) = \alpha \times WMEWMA + (1 - \alpha) \times PRR \tag{2.2}$$

$\alpha \in [0, 1]$ is the history control factor, which controls the effect of the previously estimated value on the new one. This estimator is updated at the receiver side for each w received packets.

— The RNP estimator assesses the average number of packet transmissions or retransmissions required before a successful reception (equation 2.3). Where NTRP is the number of transmitted and retransmitted packets [92].

$$RNP(w) = \frac{NTRP}{number\ \ of successfully\ \ received\ \ packets} - 1 \tag{2.3}$$

— The ETX estimator approximates the RNP. Each node explicitly broadcasts probe packets to gather statistical information. ETX takes into consideration link asymmetry by assessing the up-link quality from the sender to the receiver (PRR forward), as well as the downlink quality from the receiver to the sender (PRR backward). The combination of both PRR estimates provides an estimation of the bidirectional link quality [92].

$$ETX(w) = \frac{1}{PRR_{forword} \times PRR_{backword}} \tag{2.4}$$

	Monitoring type	**Location**
PRR	passive	receiver
WMEWMA	passive	receiver
RNP	passive	sender
ETX	active	receiver

Table 2.3: Characteristics of software-based LQEs.

The applications in industrial automation are extremely performance-sensitive i.e. these applications require reliable, timely, and secure delivery of packets at low power [6]. The TSCH dedicated to these applications stems from Time Synchronized Mesh protocols (TSMP). TSMP are network-wide time synch which are routing based with a multi-layer security on every packet. To efficiently exploit the system's performance, it uses the channel hopping technique. It also uses dedicated slotted unicast communication bandwidth and link-layer ACKs [94] [95]. Despite an acceptable performance reached due to channel hopping process, blacklisting the poorest channels may improve the performance by avoiding the failed transmissions on links that experience external interference and multi-path fading [96].

The authors in [97] proposed a lightweight strategy based on heuristics for channel selection. This strategy is designed to be implemented in cognitive and spectrum agile networks. The proposed solution chooses a small pool of channels to avoid scanning all the channels to achieve lower latency and conserve higher energy. It also dynamically refreshes the channels in the pool when the quality deteriorates by keeping higher quality channels in to efficiently exploit the network resources. The channels' quality is evaluated by testing the channel state (e.g., idle). Based on these tests, a weight is assigned to each channel. The empirical results show that this strategy achieves

high performance in terms of throughput and success ratio in the presence of cyclic, permanent, and random interference. The short coming of this strategy is that it does not guarantee convergence.

In [14] the authors proposed a new distributed approach for industrial WSN for channel quality monitoring. They proposed dedicated nodes to monitor channel quality, by using the Received Signal Strength Indication (RSSI) and the Link Quality Indicator (LQI). The experiments were performed in a real industrial environment to identify the relation between RSSI and LQI traces, and the Packet Error Rate for different channels, by using IEEE 802.15.4 radios operating in the 2.4 GHz band. The authors are able to identify the multi-path problems using only the RSSI.

In [98], the authors proposed adaptive channel hopping technique for WSN that relies on noise floor measurements to decide which channels to blacklist. Specifically, communication between nodes is periodically suspended to acquire noise floor readings. The authors proposed an Expected Blacklisting Count (EBC), an indicator of how often the channel is blacklisted. The EBC is introduced to define the average number of times that channel C is blacklisted.

In [99], the authors proposed a new channel hopping mechanism based on a cognitive radio technique to mitigate the interference between IEEE 802.15.4 and other wireless technologies in a beacon-enabled network with a star topology. A slot frame is composed of two periods, active period and inactive period. The devices can only communicate during the active period. The switching mechanism is composed of 4 steps. In the first step, the PAN coordinator uses the LQI measurement to detect whether other networks share the same channel or not. In the second step, the PAN coordinator remains alive during the inactive period of the superframe and uses the Energy Detection (ED) measurement to check whether other networks co-exist in the same channel or not. In step 3, based on measurement established during step 2, when the coordinator detects that other networks co-exist in the current channel, it uses the beacon frame to inform other devices so they switch to the next channel. During step 4, when a device notices that there is a co-existence of other networks in the current channel, it switches to a candidate channel. The channel switching time has been calculated. Theoretical analysis proves that the proposal has a short switching time and it achieves a high gain when the Beacon Interval (BI) length decreases.The proposed

work presents a trade off between accuracy in estimating a channel and exploiting the channel resources.

In [94] the authors proposed an Adaptive Channel Quality Estimation Method for Enhanced Time Slotted Channel Hoping on Wireless Sensor network. The proposed method uses Energy Detection (ED) and the simple exponential smoothing technique to estimate channel quality. It defines the interference dynamicity (ID) metric to determine whether the dynamic interference are rare or severe. Using this metric, the method improves the network's reliability and prevents energy drain caused by frequent ED. When the channel quality is highly dynamic, the method increases the frequency of ED; otherwise, it decreases the frequency of ED to gain energy.

2.8 Conclusion

In this chapter, we discuss the important techniques that overcome the common problems that occur on Cloud computing servers and IoT applications based on Wireless Sensor networks. We also present the related work that treat such problems. Next chapter presents our proposed solutions that overcome the Cloud servers challenges. It illustrates three contributions that aim to realize load balancing, fault tolerance and minimize critical and non critical requests latency time.

PART TWO: CONTRIBUTIONS

Chapter 3: Intelligent Storage in IoT-Cloud Contributions

Chapter 4: Improvement of Time Slotted Channel Hopping (TSCH)

Dealing with a huge amount of data generated by Wireless Sensors open so many issues. Among them are energy wastage, input and output response time and data availability while fulfilling real-time constraints. To deal with such problems, chapter three presents three contributions, that investigates data and data application's types to further improve Big Data structuring to address the inefficient storage, input/output response time challenges and achieve a fault tolerant and resilient system with minimum resources exploitation. Retransmitting the packets randomly risks the packets retransmission to fail several times, which influences negatively the sensor energy and packet latency. To ensure internal collision avoidance in the presence of hidden nodes, and to ensure external collision avoidance in the presence of the coexistence of other wireless technologies, chapter four presents three intelligent algorithms to minimize the latency and the network congestion, to increase the network lifetime and critical event packets lifetime, and, to ensure collision avoidance and reliability.

Chapter 3

Intelligent Storage in IoT-Cloud Contributions

3.1 Introduction

Smart objects generate enormous amount of data, called big data [4] [25]. Big data needs smart, efficient, massive storage, complex computations to extract knowledge correctly, and tremendous processing speed to enable real time decision making [4] [25]. Obviously, the used hardware and software tools do not have enough capacity to host and handle such a large amount of data within a desired period of time [25]. Sensors and actuators suffer from the restriction of physical capacities: memory, energy, and processing [2]. For such problems, the Cloud Computing is considered as a suitable solution as it has the virtual infrastructure that hosts the needed tools for the storage, and processing of big data in real time [1] [4] [25]. Effectively, it enables the emergence of new vital technologies. It is one of the essential technologies that contributes to the deployment of a successful Internet of Thing [4] [20]. One of the main utilities of the distributed servers is to provide better accessibility [43]. Unfortunately, achieving such a quality service becomes difficult when the amount of handled data is rapidly increasing. The main problem resides in the response time of recovering the critical data stored in the Cloud. The latency time will increase in case the big data storage is not efficiently organized. The absence of a tight organization of the data storage results, in the long run, in overloading some servers, in terms of CPU and storage, while other servers remain under utilized. Load balancing is an optimization technique

that seeks to distribute the load evenly among the servers in order to ensure that the sensed data is easily available on demand to users [53] [100] . Thus, it reduces latency and increases the throughput. To always provide a high data availability in critical applications, an efficient data monitoring, data backup and recovery services are highly required. Because IoT covers many critical and normal applications that generate a huge amount of data, then critical data should not be altered or deleted from the Cloud computing. For that reason, data replication is necessary to ensure data availability. On the other hand, a blind massive data replication may cause storage resources wastage.

We proposed in this chapter three contributions that address these challenges.

— The first contribution is an attempt to build an Infrastructure as a Service (IaaS), a specialization based contribution that aims to optimize the use of the servers, the architecture is called Load Balancing in the Cloud using Specialization (LBCS). The main aim of LBCS is to minimize the client's latency time. It is based on the parallelism and the cooperation concepts. Hence, this part suggests a solution, based on the paradigm of multi-agent systems to solve this complex problem. We have also applied the Contract-Net protocol to achieve accuracy in the task assigned [7].

— The second contribution called, ROBUST (Reliable lOad Balancing Using Specialization for IoT critical application) presents an Infrastructure as a Service (IaaS), an enhancement of the IaaS presented in [7] which is also based on multi-agent system. Agents are based on collaboration and parallelism to efficiently achieve load balancing among servers. The IaaS minimizes the input and the output latency time of a request in case of a massive data storage and a huge number of servers. It also serves the critical applications by ensuring a new critical data availability and replicated data recovery system. The system intelligently stores data according to their types to recover critical data rapidly in case of a server failure. Classifying data according to their types helps to duplicate critical data with minimum storage resources exploitation [101].

— The third contribution is an improvement of the idea presented in [101]. It evaluates the performance of ROBUST and LBCS by calculating the latency of critical requests when all servers are working properly. It also estimates the

latency time of replicated data recovery in case of a server failure. The IaaS of this contribution addresses the problem of single point failure that ROBUST suffers. Our algorithm is implemented using JADE Platform (Java). It has been tested with different number of output critical requests.

3.2 First Contribution: Load Balancing in the Cloud Using Specialization (LBCS)

This section illustrates our objectives and the metrics considered to achieve efficient load balancing, presents in detail our proposed solution, discusses the classic architecture, and analyses the main results of the latency time obtained while implementing the two architectures.

3.2.1 The physical architecture

We proposed an architecture which is suitable for a very large and complex network. In our architecture, we classified the servers depending on their storage capacity. For example, the cluster that groups the servers with largest storage capacity hosts the video files type, because this type requires a lot of space. The second cluster that groups servers with low storage capacity is used to store files of type image. The third one that groups the medium storage capacity servers is used for other files types, like text, XML, etc. Each cluster will have a cluster head which assigns the clients' requests to the appropriate server in its cluster. Also, the whole model contains an intermediate server between the clients and the servers. It is the main controller which delegates the client's request to the appropriate cluster head (**see figure** 3.1).

1. The client sends a request to the resource manager (as an example: the request of a video file).
2. The resource manager identifies the type of the requested file "video,image or other (text,XML,etc..)".
3. The resource manager forwards the request to the appropriate cluster head.
4. The concerned cluster head searches in its cluster the appropriate server for the assignment of the task.

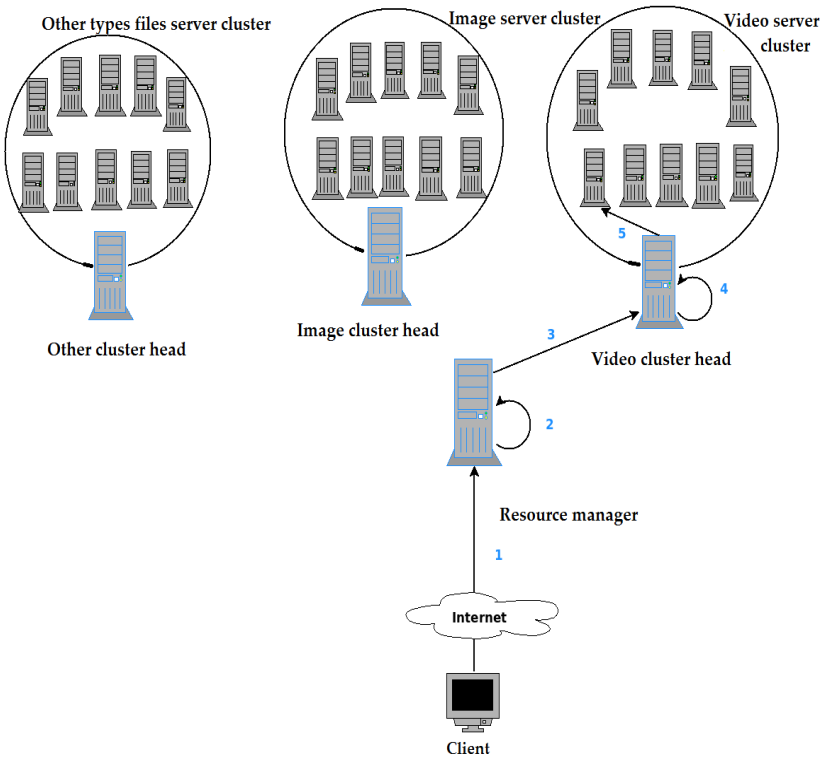

Figure 3.1: Illustration of a centralized hierarchical in a large and complex files system.

5. The task assignment.

3.2.2 The task assignment Process

Since the appropriate resource must be used, an efficient load balancing protocol must be apply. To achieve this exploitation of the resources, we focused on the accuracy of the incoming task assignment to the appropriate server, whether in the case of the input or the output of a file. In the case of input, we proposed to choose the underloaded site in terms of storage capacity. We can distinguish if a server is in an overload or in an underloaded state by a determinate threshold which differs from one site to another [52]. In this case, we have to collect the accurate state load of each server. When we have critical information, redundancy is important. In case, many sites have the required file (XML/ text, video, or image), we select the underloaded server in terms of the CPU workload.

3.2.3 Load Balancing Technique

3.2.3.1 The dynamic approach

For accuracy purposes in choosing the best destination, we apply a dynamic approach. This approach is based on the new updated information, in other words the current load state information of the system is required. The previous knowledge are not important. The advantage of the dynamic approach is that it supports the system flexibility [54] [100]. Hence our proposed load balancing algorithm collects the CPU load, the storage load state, and the database information during the execution time. Hence, this operation gathers the new updated information to guarantee a high throughput and an optimal resource use. Another reason for using a dynamic approach is to avoid beforehand the problem of the overloaded sites and unbalanced use of the resources at the servers. It also minimizes the amount of the overhead (the movement of tasks during inter-process and inter-processor communication) caused by the unbalanced use of resources.

When the dynamic approach checks a large number of servers, it increases the latency time as well as the amount of overhead. For that reason, we tried to minimize as much as possible the number of servers to be checked during the incoming of the clients' requests.

3.2.3.1.1 The minimization of the overhead: When using the dynamic approach, whenever a clients' request occurs, the appropriate cluster head sends a request to all its servers asking about their new CPU, storage load state, and their database information. The requested servers respond by sending the required information. The cluster head selects the most suitable of these servers to assign the required request.

The drawback of this approach is that it takes too much time for the cluster head to request all its servers. There are servers which do not need to be requested since their database is unchanged. To minimize the number of the messages circulating on the network (minimizing the overhead) and to reduce the cluster head CPU load, we modified this approach as follows:

Initially, the cluster head requests all the servers that are members of its cluster. After it receives the required information, it saves this information. This request is sent just once. After that, if the database of one of these servers is updated, this server (that

hosts the updated database) will immediately send the new database information and the load information to its appropriate cluster head. In this case, the servers with the updated databases send their information to the appropriate cluster head, the other servers with the unchanged databases do not send anything since the cluster head already have the information.

To enable the cluster head to know that a new updated information from a server has arrived. Before a server sends a new update of its database, it must increase its sequence number, then it sends the new sequence number and the database. The cluster head compares the new sequence number with the old one. If it finds that the new sequence number is greater than the old one, then it replaces the old database information with the new one; otherwise, the old database information remains unchanged.

When the resource manager delegates the output file request to the appropriate cluster head, this latter reads the client's request (the request contains all the information that identifies this file: the ID of the sensor that captured this data, the position of this sensor, the name of the file and the necessary information of the administrator of this application). Afterward, the cluster head surfs its directory to request the servers that host the required file. After it receives the required information from the requested servers, it assigns the request to the server with the underloaded CPU. In the case of an input file request, the cluster head, assigns the request to the server with the most underloaded storage.

This approach decreases the response time because of the minimization of the interrogated servers. Also, this method checks if the appropriate server is not in a failure state (**see figure 3.2**).

1. Client sends a video file request to the resource manager.
2. After diagnosing the requested file type, the resource manager delegates the request to the appropriate cluster head.
3. The cluster head finds out the servers that have the requested file.
4. The concerned cluster head checks again the state load of each server to select the appropriate one.
5. Task assignment.

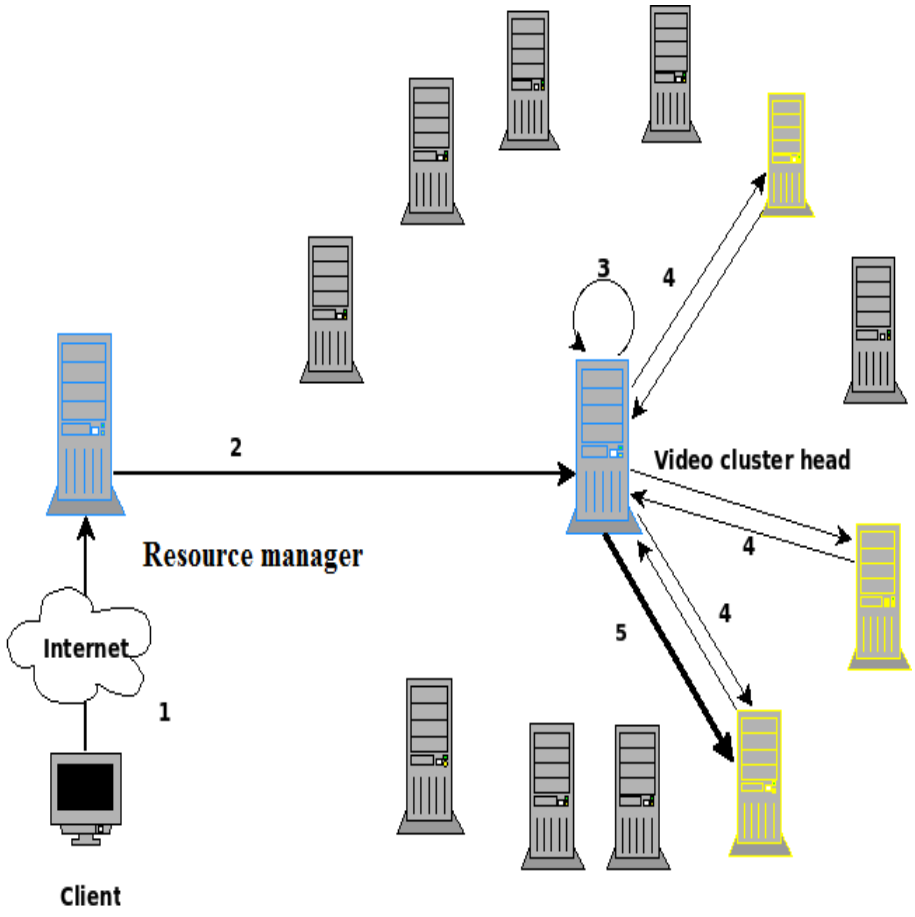

Figure 3.2: Checking the load state information of the sub-set servers that host the requested file

3.2.3.2 Contract-Net Protocol

According to [102], when a site becomes overloaded, it requests a set of servers selected randomly. The set of servers which are concerned by this request respond by an offer. Once the overloaded site chooses the best offer, the load transfer between them begins. The feature of the offer on which the sender relies to choose the best site is the most underloaded server.

The drawback of this algorithm is that when an overloaded server selects randomly a set of servers, these servers may not be the appropriate ones in terms of the storage and the CPU load state, whereas, many other servers located in the same cluster are underloaded. In this case, the overloaded server does not properly exploit the system's resources.

3.2.3.3 Contract-Net protocol adapted:

We ameliorated this algorithm by the elimination of the random selection of the set servers. In this case, the set servers are selected depending on the information collected by the cluster head on the incoming of the client's requests (as it is illustrated in **figure 3.2**).

3.2.4 Multi-agents system

We implemented our method using multi-agent systems as follows:

— One agent called **Principal-Agent**. This agent is located in the resource manager server.

— Two agents in each cluster head (The **Agent-Video-Supervisor** and the **Agent-Video-Annuaire** in the video cluster head. The **Agent-Image-Supervisor** and the **Agent-Image-Annuaire** in the image cluster head. The **Agent-Other-Supervisor** and the **Agent-Other-Annuaire** in the other cluster head.)

— Three agents called **Agent-Surveillance**, **Agent-Monitor** and **Agent-Supervisor**. These agents are located in all servers within all clusters. **See figure** 3.3.

3.2.4.1 Agents in the resource manager

3.2.4.1.1 Principal-Agent: It delegates the requests to the appropriate cluster head **(See Principal-Agent script)**.

3.2.4.2 Agents in the cluster head

3.2.4.2.1 Agent-Video-Supervisor: This agent communicates with the Principal-Agent to recover the list of the requests and sends them to the Agent-Video-Annuaire. **(See Agent-Video-Supervisor script)**.

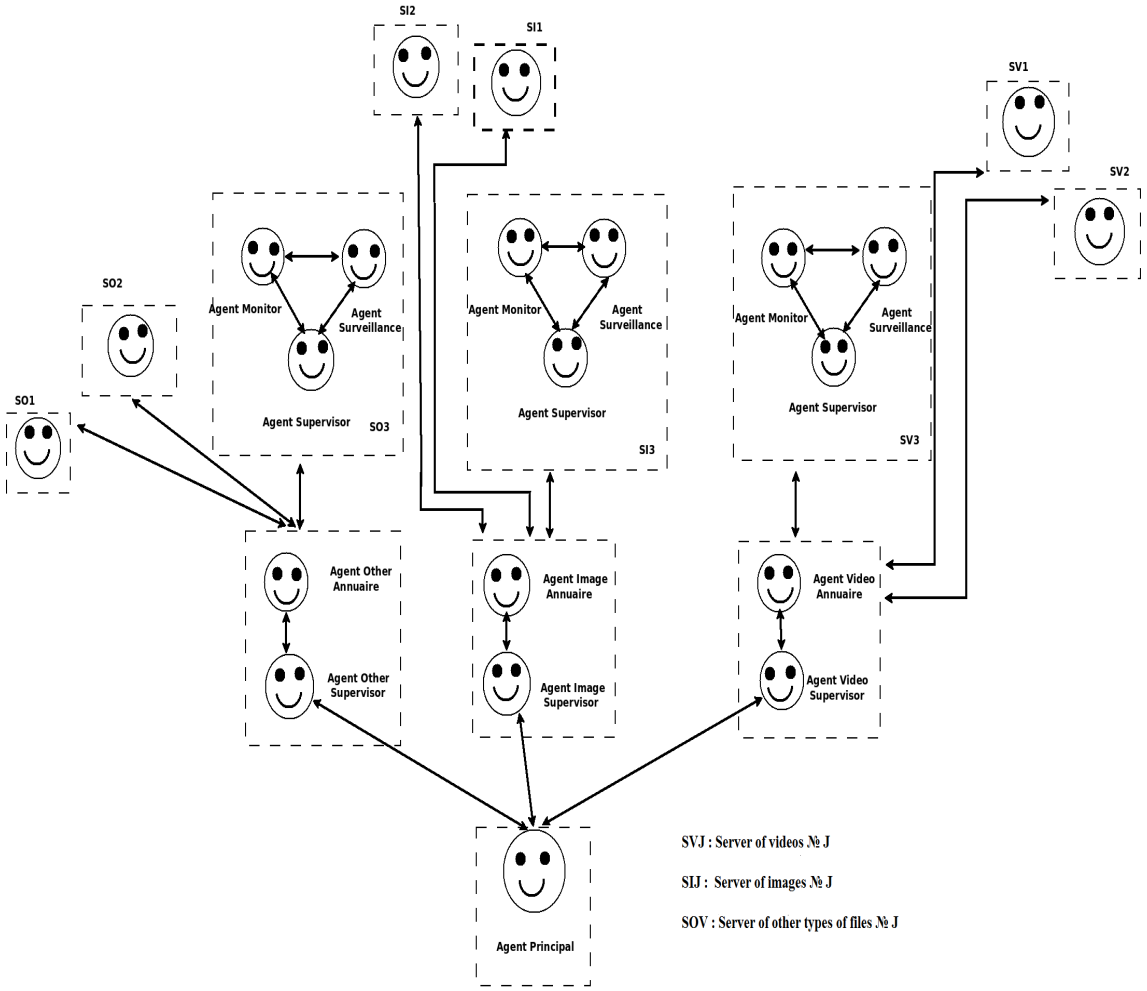

Figure 3.3: Identification of different agents in our system

3.2.4.2.2 Agent-Video-Annuaire: This agent receives the requests sent by the Agent-Video-Supervisor. It recovers periodically the CPU load status, the storage load state and the database information of the servers to assign the request to the appropriate server. **(See Agent-Video-Annuaire script)**.

3.2.4.3 Agents on the servers

3.2.4.3.1 Agent-Monitor: For each database update, the Agent-Monitor informs the Agent-Surveillance about the changes that occur in the database of this server. It also sends the new database information and the load state information to the Agent-Supervisor.

3.2.4.3.2 Agent-Surveillance: After this agent recognizes the changes that occur in the database, it increments a sequence number and sends it to the Agent-Supervisor.

Algorithm 2 Principal-Agent script

```
BEGIN
if I receive (client-Input/Output-request, name-file) then
  if (the requested file is a video type) then
    Send (client-Input/Output-request, name-file) TO Agent-Video-Supervisor
  end if
  if (the requested file is image type) then
    Send (client-Input/Output-request, name-file) TO Agent-Image-Supervisor
  end if
  if (the requested file type is neither video nor image ) then
    Send (client-Input/Output-request, name-file) TO Agent-Other-Supervisor
  end if
end if
END
```

Algorithm 3 Agent-Video-Supervisor script

```
if I receive (client-Input-request, name-file) from Principal-Agent then
  Send (client-Input-request,name-file) To Agent-Video-Annuaire
else
  if I receive (client-Output-request, name-file) from Principal-Agent then
    Send (client-Output-request,name-file) To Agent-Video-Annuaire
  end if
end if
```

The incremented sequence number denotes a new update of the database **(See Agent-Surveillance script) (see algorithm 5).**

The Agent-surveillance sends periodically a message 'Ping' to check whether Agent-Monitor is functioning or not. In case Agent-Monitor is not functioning, Agent-surveillance sends immediately an alert message to the system administrator who will fix the problem.

3.2.4.3.3 Agent-Supervisor: For each database update, this agent receives the database information and the load state information from the Agent-Monitor and, the new sequence number from the Agent-Surveillance. Once the Agent-Supervisor receives this information, it forwards it to the Agent-Video-Annuaire to update its directory **(See Agent-Supervisor script)**.

3.2.5 Implementation and results

We have implemented our new four-level architecture: clients, resource manager, cluster head, and servers. We have also implemented the process followed by the

Algorithm 4 Agent-Video-Annuaire script

```
BEGIN
% Seq-new is the new sequence number received by the Agent-Supervisor
% Seq-old is the old sequence number stocked in the Agent-Video-Annuaire during
the last update
% Request-information-msg is a request sent to a server to recover the state of its
(CPU, disc and the database information).
% —Before the incoming of the clients' requests—
for < i := 1 to number-of-servers > do
  Send (Request-information-msg) TO Agent-Supervisor[i]
  if I receive (Response-information-msg) From Agent-Supervisor[i] then
    Update the information of the server[i]
  end if
end for
% —During the incoming of the client's request—
if I receive (client-Output-request, name-file) FROM Agent-Video-Supervisor then
  select from my directory the subset servers that host the required file and stock
  them in the table sub[]
else
  if I receive (client-Input-request, name-file) FROM Agent-Video-Supervisor then
    select the most underloaded subset servers that lack the required file and stock
    them in the table sub[]
  end if
  for < i := 1 To sub.length > do
    Send(Request-information-msg) TO sub.Agent-Supervisor[i]
    if I receive (Response-information-msg) From sub.Agent-Supervisor[i] then
      if server[i].Seq-new > server[i].Seq-old then
        Update the information of the server[i]
      else
        if server[i].Seq-new == server[i].Seq-old then
          DO NOTHING ;
        end if
      end if
    end if
  end for
  if (client-Input-request) then
    Select from the table sub the most suitable server in terms of the storage load
    state.
    Send (client-Input-request, name-file) TO Agent-supervisor of the appropriate
    server
  else
    Select from the table sub the appropriate server in terms of the CPU load state.

    Send (client-Output-request, name-file) TO Agent-supervisor of the appropriate
    server
  end if
end if
END
```

Algorithm 5 Agent-Surveillance script

```
if I receive (Response-information-message) From Agent-Monitor then
  sequence-number ++
  Seq-New := sequence-number
  Send (Seq-New) To Agent-Supervisor
end if
```

Algorithm 6 Agent-Supervisor script

```
Begin
% The send of the updated information
for < i := 1 to number-of-servers > do
  if (I receive (Seq-New) From Agent-Surveillance AND I receive (new-database
  information) From the Agent-Monitor) then
    Send(Seq-New,new-database information,Server[i]) TO Agent-Video-Annuaire
  end if
end for
End
```

delegation of the clients' requests without using the notion of the specialization [7]. Unlike our proposed solution, the second approach contains just two levels which are: the clients and the servers that host the database and respond to the input and the output requests. In other words, all servers may host all file types. We mention here that there is no intermediate server between the clients and the servers that host the database. There is neither a resource manager nor a cluster head. The result of the comparison is illustrated in figures 3.4 and 3.5. It shows that there is a significant improvement of latency time.

Figure 3.4 shows the behavior of our system for the case of output files. We noticed that in a non specialization architecture when the number of requests increases, the response time increases very rapidly (requests between 30 and 40). But in our proposed architecture we notice that when the number of requests increases, the response time increases also but in a reasonable way. For the case of 40 requests, the difference of the response time between the two architectures is equal to 861 ms.

Figure 3.5 shows the gain (improvement) of the time measured for the case of input files. We notice that there is a clear improvement in the latency time. For example, for the case of 40 requests we noticed an improvement of 759 ms.

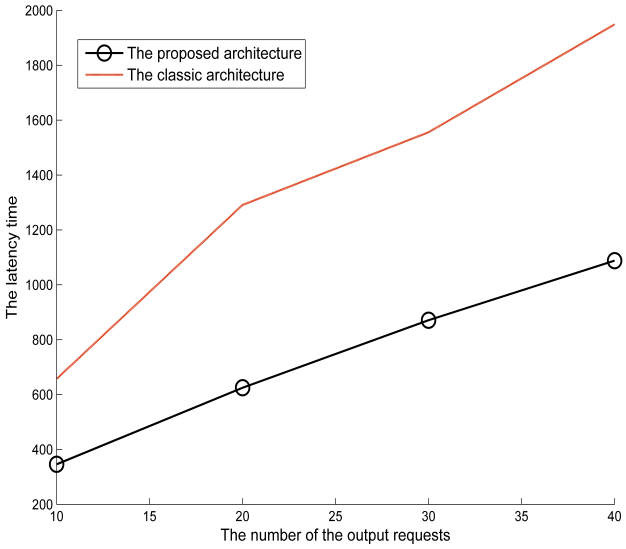

Figure 3.4: Evaluation of the latency time according to the number of the output requests

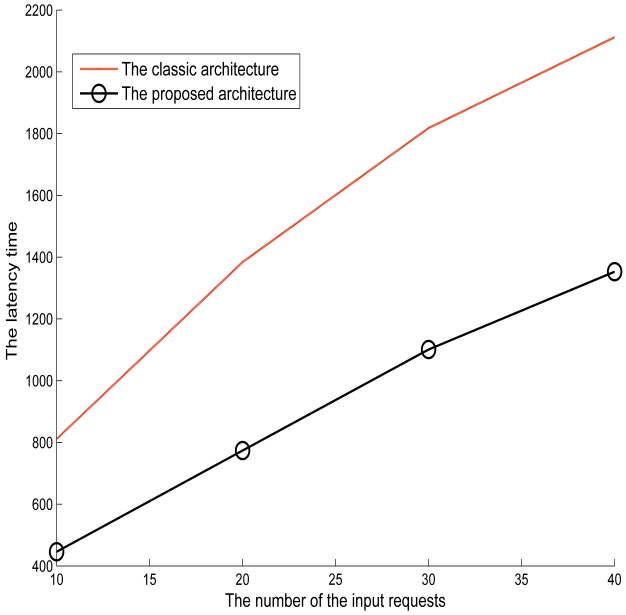

Figure 3.5: Evaluation of the latency time according to the number of the input requests

3.3 Second Contribution: A new Infrastructure as a Service in IoT-Cloud

This section introduces the aim of the work and the Cloud architecture, explains the Multi-agent system, discusses the obtained latency using the complexity cost, and expalins the system monitoring, backup and recovery services, and the systems' per-

formance.

3.3.1 The aim of this work

In this work we create a new IaaS based on the collaboration and the parallelism of multi-agent system. This IaaS is an enhancement of the IaaS presented in [7]. It aims to:

— spread tasks evenly among the servers to achieve better load balancing in the Cloud,

— minimize the latency time of critical requests,

— increase the throughput (the number of serviced critical requests per unit of time).

The idea consists of classifying the Cloud servers into three clusters [7].

Each cluster$_i$ contains two sub-clusters. Each sub-cluster stores the sensed data according to the sensitivity degree of the application. We classify the applications into two main classes: critical applications and non critical applications. The critical applications are those that should strictly respect the real time constraints, and all the previous (stored) sensed data is needed to make future decisions. Also the applications related to human life are considered critical (e.g., earthquake, patient health-care, and security etc..). The non critical applications do not need to satisfy the real time constraints (e.g., the application related to social information etc...) or ones whose damaged/lost stored data does not influence future decisions.

3.3.2 Multi-Agents system

We implemented our method using multi-agent system as follows:

— One agent called **Principal-Agent**. This agent is located on the resource manager server.

— Three agents in each cluster head (The **Agent-Video-Supervisor**, **V-crit-req-Agent V-req-Agent**) in the video cluster head (CH-V)).
(The **Agent-Image-Supervisor**, **I-crit-req-Agent I-req-Agent**) in the Image cluster head (CH-I)).

(The **Agent-Other-Supervisor**, **O-crit-req-Agent O-req-Agent**) in the Other cluster head (CH-O)).

— Two agents located on each server in the cluster (**Agent-D** and **Agent-surv**).

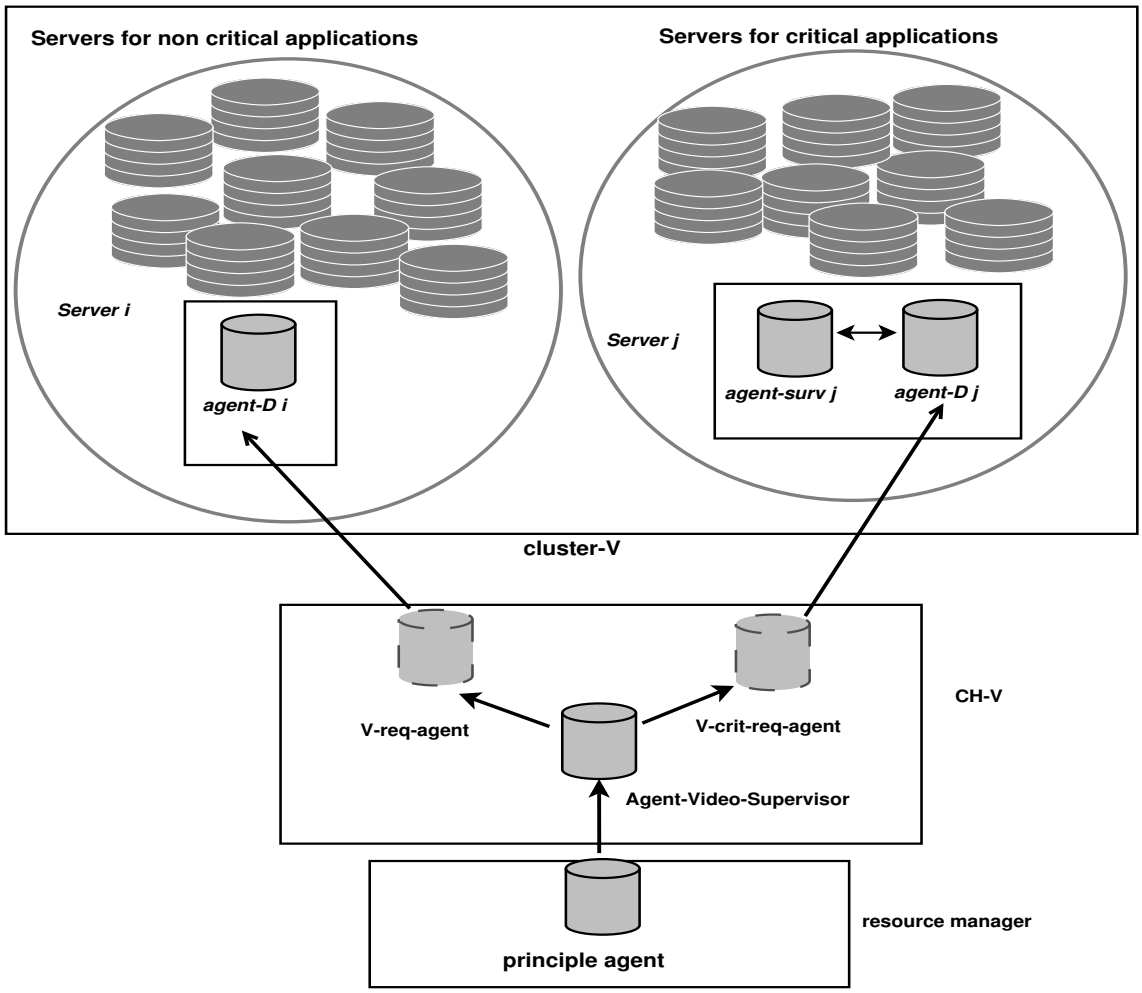

Figure 3.6: Identification of different agents in our system

Figure 3.6 shows the different agents in our system (video servers)

3.3.2.1 Agents in the resource manager

3.3.2.1.1 Principal-Agent: It forwards the requests to the appropriate cluster head. The steps are detailed in [7].

3.3.2.2 Agents in the cluster head

3.3.2.2.1 Agent-Video-Supervisor: This agent communicates with two agents located in the same cluster head (CH-V). If the request belongs to a critical application,

then it forwards the request to the V-crit-req-agent; otherwise it forwards it to the V-req-agent.

Algorithm 7 Agent-Video-Supervisor script

```
BEGIN
if I receive (client-input/output-crit-request, name-file, ID-app) from Principal-Agent then
   Send (client-input/output-crit-request, name-file, ID-app) to V-crit-req-Agent
else if I receive (client-input/output-request, name-file, ID-app)from Principal-Agent then
   Send (client-input/output-request, name-file, ID-app) to V-req-Agent
end if
END
```

3.3.2.2.2 V-crit-req-Agent: It receives the requests from the Agent-Video-Supervisor. It also recovers periodically the CPU load status, the storage load state and the database information from servers that store critical data. When an input request arrives, it assigns it to the underloaded server in terms of storage load state to ensure load balancing among servers (see Algorithm 8).

3.3.2.2.3 V-req-Agent: It receives the requests sent by the Agent-Video-Supervisor. It tries to ensure load balancing among servers that belong to the second sub-cluster (the sub-cluster that contains servers for non critical applications).

3.3.2.3 Agents on servers

3.3.2.3.1 Agent-D: Is an agent deployed on each server. It hosts the database of the server. It is responsible for the delivery and the storage of the sensed data.

3.3.2.3.2 Agent-surv: This agent is responsible for monitoring the state of its neighbor.

3.3.3 Latency estimation

In this section, we estimate the latency of a requested file. To show the performance and illustrate the enhancement of our new IaaS, we compared the latency time of

Algorithm 8 V-crit-req-Agent

```
BEGIN
% number-of-servers-CA is the number of servers that hold the critical applications data
% Request-information-msg is a request sent to a server to recover the state of its (CPU, disc and the database information).
% —Before the incoming of the clients' requests—
for < i := 1 to number-of-servers-CA> do
  Send (Request-information-msg) to Agent-D
  if I receive (Response-information-msg) from Agent-D[i] then
    Update the information of the server[i]
  end if
end for
% —During the incoming of the client's request—
if I receive (client-output-request-CA, name-file) FROM Agent-Video-Supervisor then
  select from my directory the subset servers that host the required file and stock them in the table sub[]
else
  if I receive (client-input-request-CA, name-file) FROM Agent-Video-Supervisor then
    select the most underloaded subset servers that lack the required file and stock them in the table sub[]
  end if
  if (client-input-request-CA) then
    Select from the table sub the most underloaded server in terms of the storage load state.
    Send (client-input-request-CA, name-file) to Agent-supervisor of the appropriate server
  else
    Select from the table sub the underloaded server in terms of the CPU load state.

    Send (client-output-request-CA, name-file) to Agent-supervisor of the appropriate server
  end if
end if
END
```

Algorithm 9 Agent-D script

Begin
if (I receive (Response-information-msg) from V-crit-req-Agent) **then**
Send the CPU and the storage load state, and the database information to V-crit-req-Agent.
else
if (I receive (Response-information-msg) from V-req-Agent) **then**
Send the CPU and the storage load state, and the database information to V-req-Agent.
end if
end if
End

ROBUST [101], LBCS [7] and the classic architecture [7]. The latency is calculated according to equation 3.1.

$$DT = K \times NB \times time \tag{3.1}$$

Parameters

— DT is the delivery time.

— *time:* is the necessary time for fetching a file, $time = \alpha$, where α is a unit of time.

— K is the number of servers in each sub-cluster, $K = 50$. K in LBCS is equal to 100 (each cluster contains 100 servers because of the absence of sub-clusters). K in the classic architecture is equal to 300.

— NB is the number of files on each server. We assume that the average of the total number of files equals to M.

Figure 3.7 represents the latency for an output request vs the number of files on each server using ROBUST architecture, LBCS and the classic architecture. Note that the latency of a file output request when using our architecture increases very slowly compared to the LBCS and the classic architecture. The gap between our new architecture and the 2 other architectures grows rapidly when the number of files stored on each server increases. Note that when the number of files reaches 5000, the latency gain of ROBUST compared to LBCS reaches $495 \times 10^3\alpha$ while the latency gain of our architecture compared to the classic one is $1495 \times 10^3\alpha$. This performance improvement is the result of agents' collaboration and parallelism, and servers specialization.

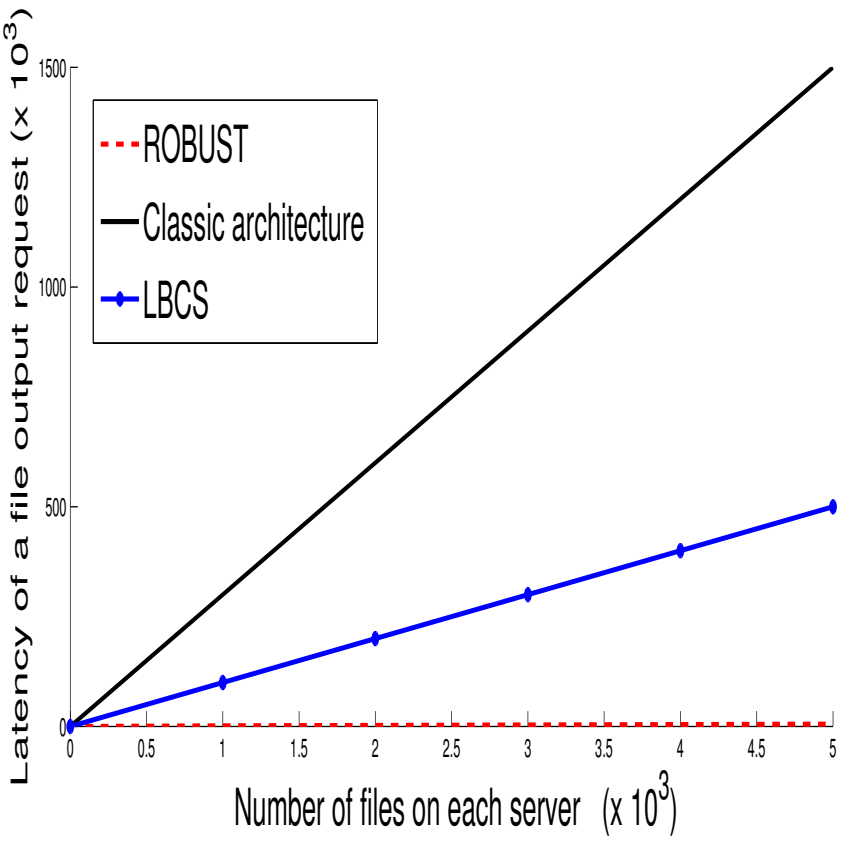

Figure 3.7: Latency for an output request vs the number of files on each server using three Architectures

3.3.4 System Monitoring, backup and recovery purposes

To achieve data availability in critical applications we propose a new mechanism based on the event triggered protocol for backup and recovery purposes. We also adapt the TTC/P to ensure a high system monitoring. The combination of these two methods is efficiently implemented by the collaboration and the parallelism of multi-agent system.

The aim of this mechanism is to:

— realize the servers' state awareness by ensuring a high monitoring,

— ensure data availability in case of the required server failure (the one on which the required data is stored),

— search the duplicated version of a data as fast as possible in a massive data storage and huge number of servers.

Assumptions:

— The clocks are always synchronized.

— All servers are load balanced (based on the existing system experience).

— At most one server failure at a time.

— Only temporary/transient failures can happen.

— The failures can happen just on servers (no links failures).

— The failure can only occur at the receiver side.

3.3.4.1 System Monitoring

To ensure an efficient system monitoring, the detection of each server's neighbor is required. This section explains how do the agents detect the neighbor of each server, and how do they achieve the system monitoring.

3.3.4.1.1 - Detecting server's neighbor : The V-crit-req-Agent statically arranges the IP address of all critical applications servers into a logical ring (e.g., the IP addresses are arranged in a round robin order). Once the list is established, the V-crit-req-Agent distributes it to all critical servers so each server$_i$ can communicate with its neighbor according to the order in the list. We mention that the neighbor of the last server is the first server.

3.3.4.1.2 - Adapting the TTC/P to ensure system monitoring : We apply the time triggered protocol (TTC/P) to constantly and safely ensure critical application monitoring. To decrease the congestion between nodes, a host detects its neighbor's failure state by unicasting a message to its neighbor. In our method, each TDMA round is splitted into N time slots, where N is the number of critical application servers. Each Agent-surv$_i$ on each server is assigned a fix and periodic duration (time slot$_i$ **mod** N) of unicast during which it can transmit a hello message to its neighbor cited in the list and receive an acknowledgment from the receiver. The hello message is a special packet (message) that is sent out periodically to test a reachability of a neighboring server. As indicated in Figure 3.8, the acknowledgement is used to interpret the life-sign of the receiver. If the Agent-surv$_i$ detects that the Agent-surv$_{i+1}$ is out of order, it immediately informs the V-crit-req-Agent about the server$_{i+1}$'s failure (see Algorithm 10).

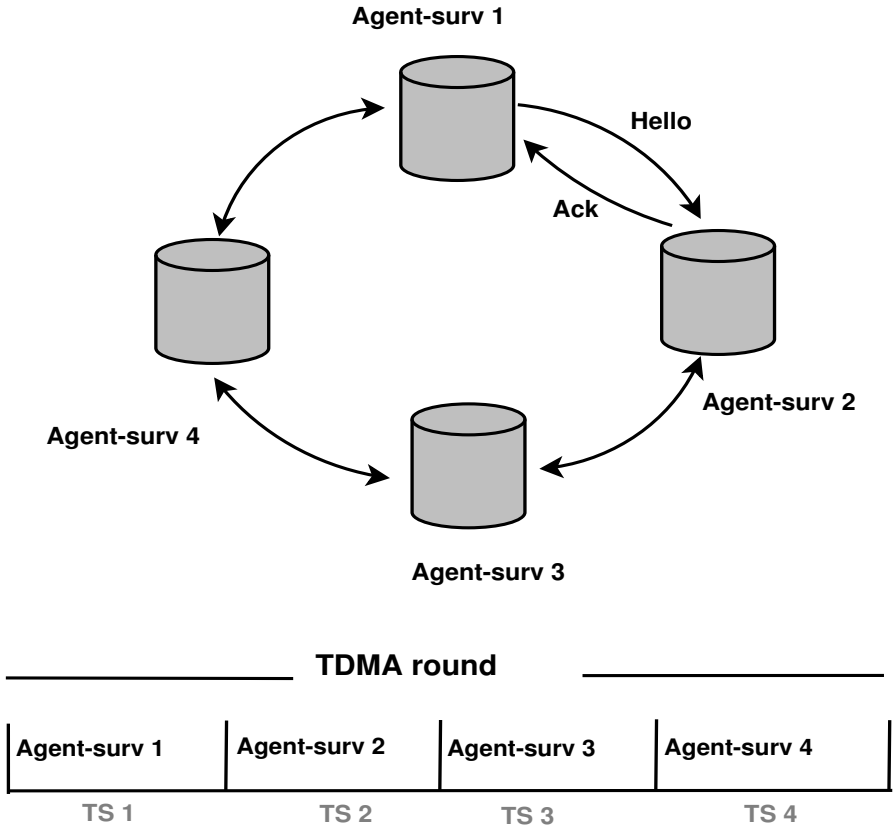

Figure 3.8: System Monitoring based on the pre-established servers order

3.3.4.2 Data replication

To provide a serious attention for ensuing data redundancy, we applied the event triggered protocol to quickly duplicate the new data as soon as the server$_i$ successfully stores it. We apply this method to avoid the risks of the server$_i$ damage/failure before the data replication.

Upon the reception of each new data, each Agent-D_i duplicates the new received data on its neighbor (see Algorithm 11).

Algorithm 10 Agent-surv$_i$

Begin
for each time slot$_i$ **do**
  Send hello to Agent-surv$_{i+1}$
  if Agent-surv$_i$ does not receive an ACK from the Agent-surv$_{i+1}$ before the time slot$_i$ elapses **then**
    Inform the V-crit-req-Agent that server$_{i+1}$ is out of order.
  end if
end for
End

Algorithm 11 Agent-D_i

Begin
if (I receive (client-input-request-CA, name-file) from (V-crit-req-Agent) **then**
send a copy of the new file to Agent-$D_{i+1 mod N}$ located on $Server_{i+1 mod N}$
end if
End

3.3.4.3 Data recovery

When the V-crit-req-agent receives an output $request_i$ stored in agent-D_i, it sends it a hello message to check whether it works properly or not. If agent-D_i replies to the V-crit-req-agent with an acknowledgement, the V-crit-req-agent assigns it the output request; otherwise, it forwards the request to the agent-D_{i+1} located on $server_i$ neighbor as shown in Figure 3.9

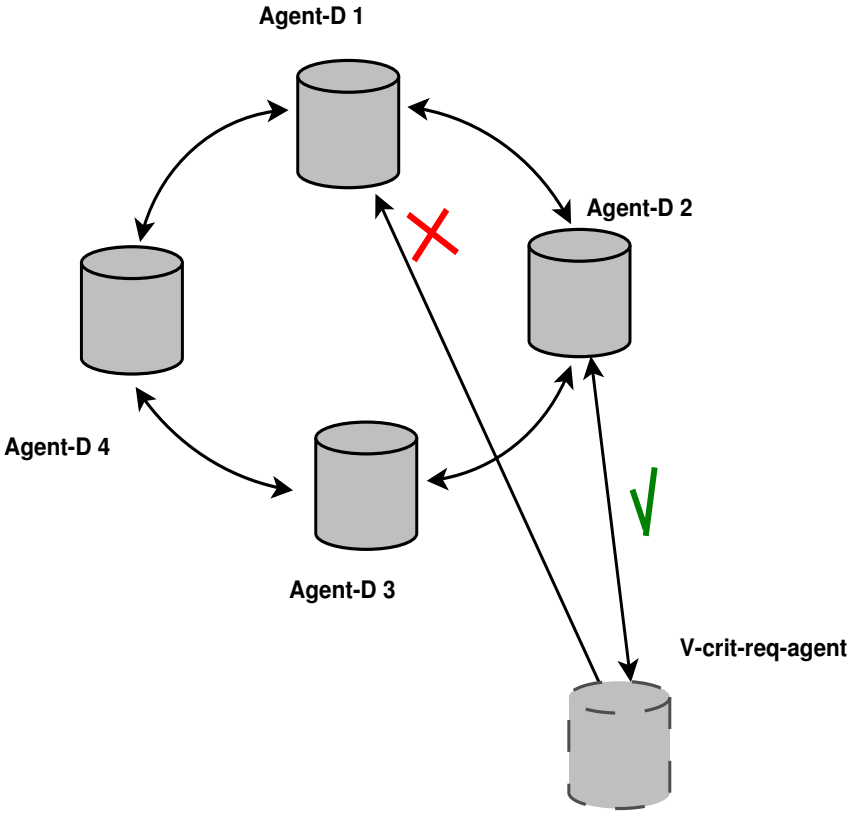

Figure 3.9: Recovery of the replicated data version

3.3.4.4 Data recovery complexity

To show the performance of ROBUST, we compare the efficiency of ROBUST with both the LBCS and the classic architecture.

3.3.4.4.1 Searching for a requested file in ROBUST If the V-crit-req-agent receives a client input request of a data stored on a failed server ($server_i$), it serves the request rapidly since it is aware about replicated data location. The V-crit-req-agent will not search all servers about the copy of the required file, it will just fetch/search $server_{i+1}$, the neighbor of $server_i$ (see Algorithm 12).

Algorithm 12 Searching for a requested file by agent-D_{i+1}

```
Begin
if (I receive (client-output-request-CA, name-file) from (V-cri-req-agent) then
  for <i := 1 to M> do
    if M_i = name-file then
      Transmit the required file to the client.
    end if
  end for
end if
End
```

$$Complexity = 1 + M \times 2 \tag{3.2}$$

1 : is for the first instruction (IF).

M : is for the M instructions executed in the loop (FOR).

2 : is the two instructions inside the loop (FOR).

When using ROBUST, the complexity of searching for a file is **O(M)**.

3.3.4.4.2 Searching for a requested file in the LBCS and classic architecture The complexity when a server searches the required file on a failed server. Note that in the classic architecture there is no classification of servers; all servers store both critical and non critical data. When the server that hosts the requested file is out of order, the resource manager does not know the location of duplicated copy of requested file. As described by Algorithm 13 in this case the resource manager has to search all servers as long as it does not find the required file.

Parameters

— N is the number of all servers that may store the required file. In the classic architecture M = 300. In LBCS M = 100 (the research will be limited because the servers are organized into clusters)).

Algorithm 13 Searching for a requested file by agent-D$_{i+1}$

```
Begin
if (I receive (client-output-request-CA, name-file) from (V-crit-req-agent) then
  for <i := 1 to N> do
    for <i := 1 to M> do
      if M_i = name-file then
        Transmit the required file to the client.
      end if
    end for
  end for
end if
End
```

— M is the average of the total number of files stored on all servers.

$$Complexity = 1 + N \times (M \times (1 + 1)) = 1 + 2 \times N \times M \tag{3.3}$$

1 for the first instruction (IF).

M for the number of instructions executed by the loop for. It means the number of files on each servers.

N is the number of servers.

2 is the two instructions executed inside the loop for. One for testing the requested file and the other one for transmitting the request to the client.

So, the complexity in the case of LBCS and classic architecture is **O(MN)**.

3.3.4.5 Results

In this section we present the messages complexity when requesting a file using ROBUST, LBCS and the classic architecture.

Figure 3.10 shows the complexity of searching a replicated version of a requested file vs the number of files on each server when using ROBUST, the LBCS and the classic one. Note that the complexity of a file output request when using ROBUST increases very slowly compared to the LBCS and the classic architectures. The gap between ROBUST and the two other architectures grows rapidly when the number of files stored on each server increases. When the number of files reaches 5000, the difference between the complexity of ROBUST and LBCS is 990×10^3. The difference between the complexity of ROBUST and the classic one is 2990×10^3. This huge

performance improvement is the result of the intelligent storage of the replicated data achieved due to the servers' load balancing and accurate monitoring of servers.

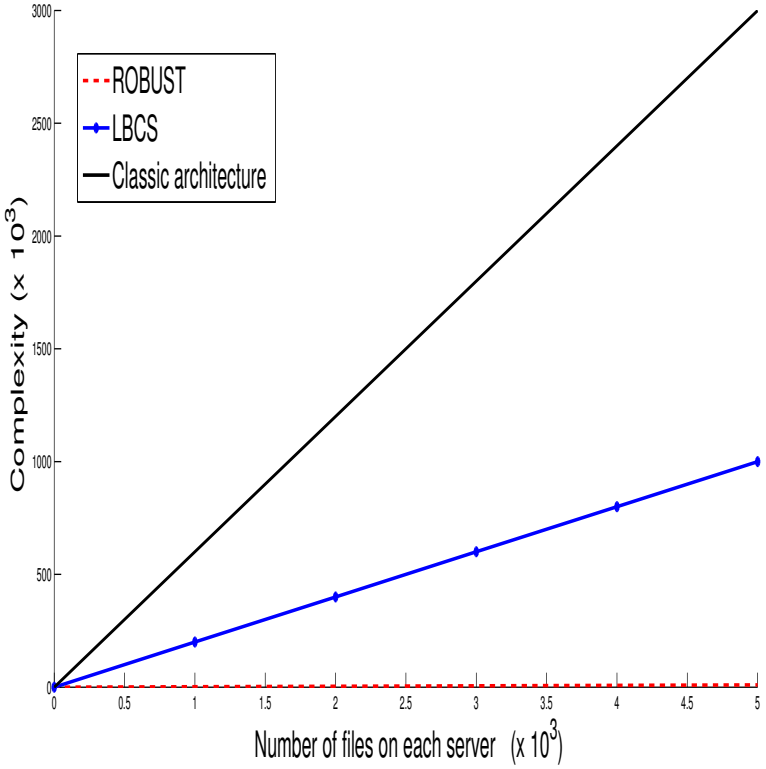

Figure 3.10: Comparison of the Complexity to Search for a file using the three Architectures

Table 3.1 shows the comparison of parameters of data replication and recovery of the three architectures.

The short latency is achieved by classifying data according to its type and applying the mechanism of data duplication which is based on round robin technique. Avoiding blind data duplication helps us to exploit more resources.

Table 3.1: Comparison of data replication and recovery parameters of three architectures

Parameter	**Classic Arch**	**LBCS**	**ROBUST**
Data availability	High	High	Hight
Latency time	High	Medium	Low
Wastage of storage resources	High	Medium	Low

3.4 Third Contribution: A Fault Tolerant and Resilient Infrastructure as a Service for Intelligent Storage in IoT-Cloud

This contribution is an improvement of the previous IaaS [101]. It evaluates the performance of ROBUST and LBCS by calculating the latency of critical requests when all servers are working properly. It also estimates the latency time of replicated data recovery in case of a server failure. It also addresses the problem of single point failure. This section explains the multi-agent systems and their characterization, evaluates ROBUST performance, explains how to avoid the single point of failure problem that may occur in ROBUST, describes data replication and recovery of ROBUST, and discusses the evaluation performance of latency time in the case of critical data recovery, and points out our main objectives, our contributions and summarizes the key results.

3.4.1 ROBUST Description

ROBUST is based on clustering intelligent servers. We used Clustering because it simply manages large and rapidly growing systems. To ensure load balancing among servers we applied a centralized approach to carefully control the system, assure ideal load balancing and achieve minimum latency to critical and real time requests.

3.4.2 Multi-Agents system

We implemented a set of agents to achieve a high reactivity and fault tolerance.

Table 3.2 presents the different intelligent agents and their characteristics in our system.

Figure 3.11 shows the different agents in our system (video servers).

3.4.2.1 NeighborV-crit-req-Agent

It seeks to ensure fault tolerant at the cluster heads servers.

Table 3.2: Identification of our system's agents their characterization

Level	Number of agents	Name of Agents	Agents type
Resource manager	one	principal-Agent	Cooperation + Transaction
Cluster heads (CH-I, CH-V, CH-O)	four	Agent-Video-Supervisor V-crit-req-Agent V-req-Agent NeighborV-crit-req	Cooperation Information + cooperation Information + cooperation Transaction + Information
Servers for critical applications	two	Agent-D Agent-surv	Information Cooperation + Transaction
Servers for normal applications	one	Agent-D	Information

3.4.3 Implementation

3.4.3.1 Assumptions

During the simulation we assumed the following:

— Servers are load balanced [7].

— Resource manager receives just video critical requests.

— All servers work properly.

Simulation parameters are shown in Table 3.3

Table 3.3: Comparison of data replication and recovery parameters of three architectures

Total Number of servers	45
Servers for critical applications in each cluster	04
Number of servers for normal applications in each cluster	11
Number of files on each server	1800
Number of Agents	100
Number of critical requests	[20, 100]

We have implemented our new four-level architecture: clients, resource manager, cluster heads , sub-cluster heads, and servers. We have also implemented the process followed by the delegation of the clients' requests without using the notion sub-clustering, where each cluster servers store both critical and normal data.

Figure 3.12 represents the number of transmitted critical video requests of both protocols vs Latency Time. Notice that in LBCS architecture, when the number of

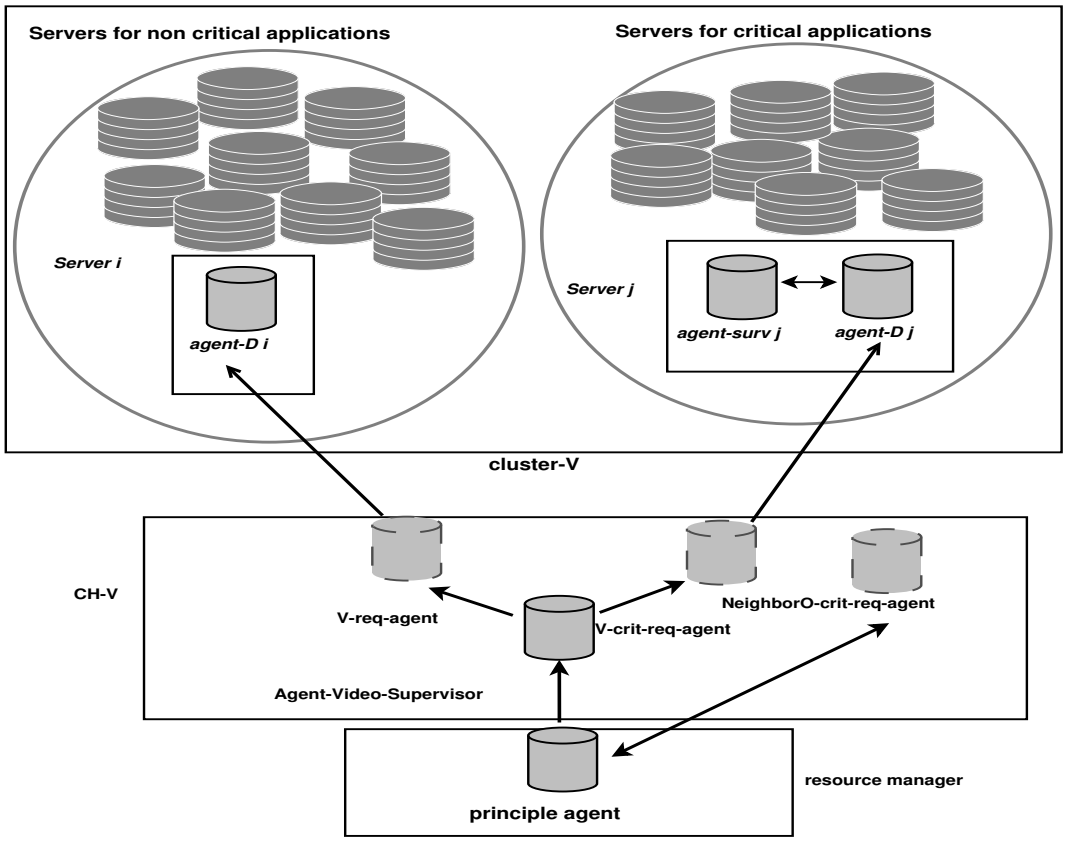

Figure 3.11: Fault tolerant MAS Architecture

critical requests increases, the latency time increases rapidly. But in ROBUST we notice that when the number of critical requests increases, the response time increases in a reasonable way. This improved behaviour is a result of the clustering that realizes parallelism, the collaboration of multi-agents system and the sub-clustering that minimized the number of servers where search for a critical data is performed.

Table 3.4 shows the elapsed latency time when sending different types (XML files, videos, and images) of requests from principle agent to Agent-supervisor.

Table 3.4: Latency estimation from principal-agent to Agent-Supervisor

Number of requests	Latency
150	24 *ms*
75	11 *ms*
30	7 *ms*

Table 3.5 illustrates the latency time elapsed when transmitting normal video requests from Agent-supervisor to V-req-agent and, critical video requests from Agent-supervisor to V-crit-req-agent.

We notice that the elapsed time when diagnosing the files types can be neglected

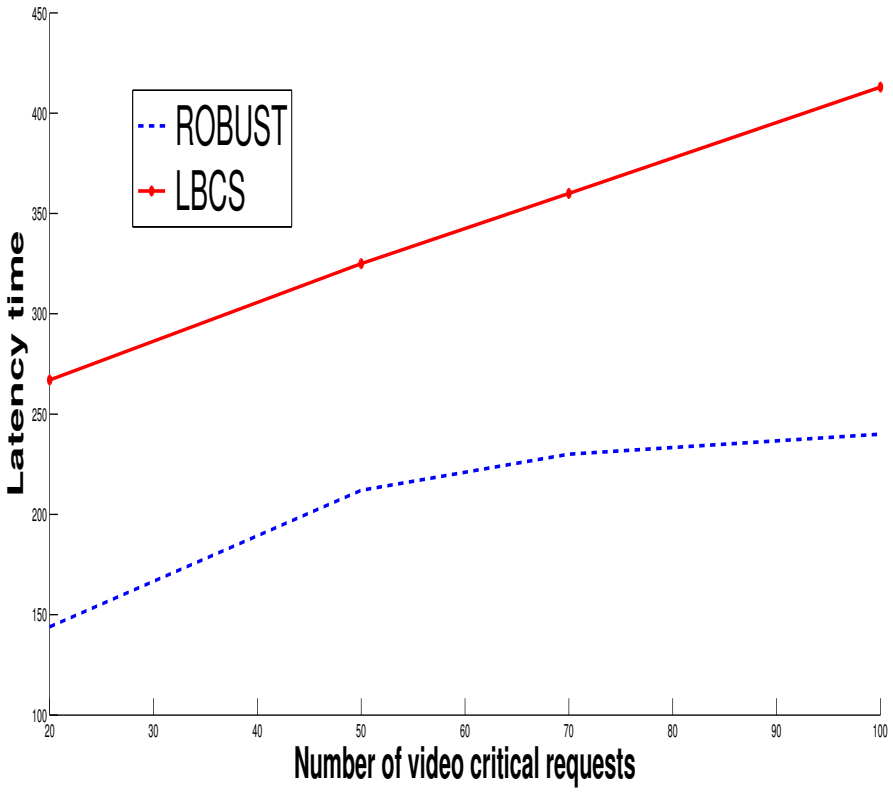

Figure 3.12: Evaluation of the latency time according to the number of the output requests

Table 3.5: Latency estimation from Agent-Supervisor to v-crit-req-Agent and from Agent-Supervisor v-req-Agent

Number of requests	Latency
50	25 *ms*
25	20 *ms*
10	9 *ms*

when we compare it with the gain of latency time achieved by using servers sub-clustering.

As expected, the centralized approach achieved a high performance in minimizing critical requests latency. The drawback of this architecture is that the cluster heads has the single point failure problem. To deal with this problem, we propose a new architecture based on redundancy.

3.4.4 Avoiding single point failure

To ensure a fault tolerant architecture, system availability and resilience while exploiting the systems resources and minimizing the bottleneck problem, we applied

Active-Standby Scenario.

Critical system information stored in the cluster head servers is duplicated using additional resources in such a way a copy of the critical system information is available even after a primary cluster head server failure happens. Duplicating is applied as follows:

principal agent arranges the IP address of the cluster-head servers (CH-I,CH-V,CH-O) into a logical ring. Once the list is established, the principal agent distributes it to all cluster head servers so each cluster head server can communicate with its neighbor according to the order in the list. We mention that the neighbor of the last server is the first server (See Figure 3.13).

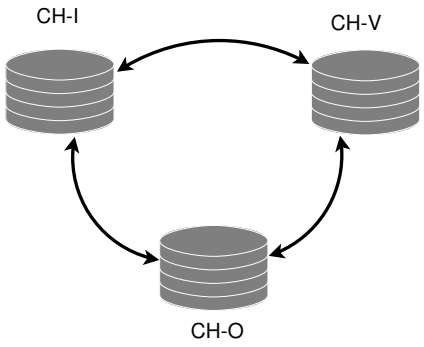

Figure 3.13: Cluster heads communication

To provide minimum latency in case of one of cluster head failure, we introduce a new agent type in each cluster head. NeighborV-cri-req-agent located in CH-I, NeighborI-cri-req-agent installed in CH-O, and NeighborO-cri-req-agent situated in CH-V. These agents store the critical applications servers information (e.g., their storage load state, last CPU load state, the list of stored files on each server in its neighbors cluster, etc...) (See Figure 3.11).

Each *i*-crit-agent sends periodically the servers information to its Neighbor *i*-cri-req-agent (See Figure 3.14).

3.4.4.1 Critical data recovery in case of a CH-V failure

Assumption

— Clocks are always synchronized.

— At most one cluster head failure at a time.

— Only temporary failures can occur.

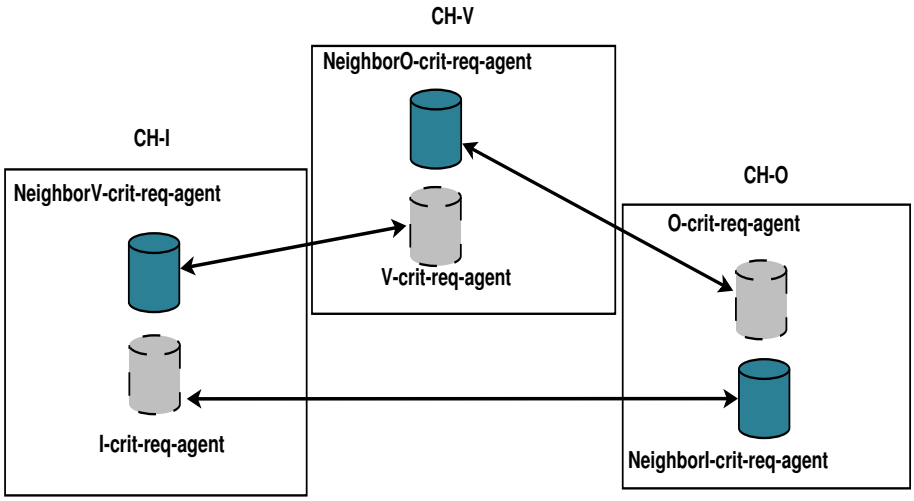

Figure 3.14: Agents in Cluster Heads

— Failures can occur just on servers, links are robust.

Each Agent-i-supervisor sends a heartbeat to the principal agent according to Time Triggered Protocol (See Figure 3.15). The hearbeat message indicates that the CH_i server works properly.

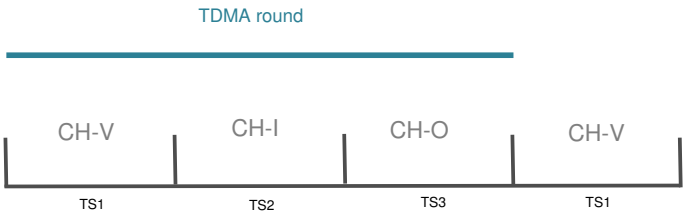

Figure 3.15: TDMA round strategy dedicated to CHs

When the principal-agent receives a client critical request (e.g., video), it forwards it to the appropriate cluster head. If the principal-agent determines that the appropriate CH_i is out of order (e.g., CH-V), it forwards the request (video critical request) to CH_i's neighbor (NeighborV-crit-req-agent situated in CH-I) (See Algorithm 14).

When a CH-$_i$ (CH-V) recovers, it sends a heartbeat to the principal agent. When the latter discovers its recovery, it asks the Neighbor i-crit-req-agent (NeighborV-crit-req-agent) to send to the i-crit-req-agent (V-crit-req-agent)the last updated critical servers information (See Algorithm 15) .

3.4.5 Performance Evaluation

Simulation parameters are shown in Table 3.6

Algorithm 14 principal Agent

```
Begin
if CH-V is out of order then
  if (I receive (video critic client-input or output-request-CA, name-file) from
  (client)) then
    Forward the request to the NeighborV-crit-req-Agent situated in the CH-I (the
    neighbor server of CH-V).
  end if
else
  if CH-V is fixed then
    if I receive a heartbeat from CH-V then
      inform NeighborV-crit-req-Agent about V-crit-req-Agent health state

    end if
  end if
end if
End
```

Table 3.6: Simulation parameters utilized when estimating latency when replicated critical data Recovery

Total Number of servers	14
critical applications Servers in cluster-V	04
Servers in cluster-V	11
Number of servers for normal applications in each cluster	11
Number of files on each server	1800
Number of Agents	34
Number of critical requests	[20, 100]

Figure 3.16 represents the number of transmitted critical video requests of both protocols vs latency time of redundant data recovery. Notice that in LBCS architecture, when the number of critical requests increases, the latency time of redundant data recovery increases rapidly, this is due to the random data duplication. But in ROBUST we notice that when the number of critical requests increases, the response time increases in a reasonable way, and this is due to the intelligent data duplication, in other words, when a primary server fails the search for the requested file is limited just on one server (primary server's neighbor).

3.5 Conclusion

Smart sensors in IoT present a storage challenge. IoT-Cloud solves such a problem since it provides users on-demand resources' access any where and any time. Assigning

Algorithm 15 NeighborV-crit-req-Agent

Begin
if CH-V is out of order **then**
if (I receive (video critic client-input or output-request-CA, name-file) from (principal Agent) **then**
Assign the request to the appropriate server.
Update the server's information (CPU, storage load, list of input files, etc...)
else
if I receive the CH-V health state from (principal Agent) **then**
Send the last updated information of video critical servers to the V-crit-req-Agent.
end if
end if
end if
End

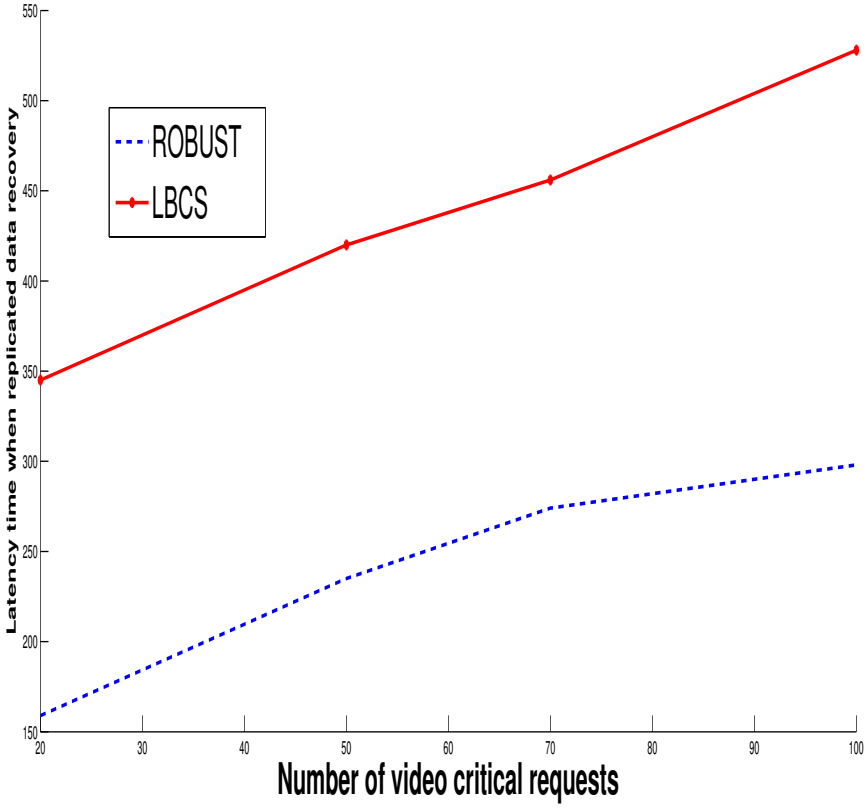

Figure 3.16: Latency time when the recovery of the replicated data

a huge number of critical and non critical requests to the servers with big data may lead, in the long run, to an unbalanced workload among the servers. This unbalance causes an increase of the latency time. This happens when the clients try to recover files. Because IoT covers many critical and normal applications that generate a huge amount of data, then critical data should not be altered or deleted from the Cloud

computing.They must be available at any time even in case of a host servers failure. As an attempt to solve these problems, we proposed in this chapter three contributions.

— In the first contribution, we created an Infrastructure as a Service (IaaS) using the notion of servers' specialization. The proposed architecture called, LBCS is suitable for a large and complex network. We implemented our architecture using multi-agent systems (we implemented a total of 26 agents). Each of these agents has its own role. While some of these agents collaborate to realize the load balancing, others work in a parallel way to achieve their tasks rapidly and minimize the client latency time. This approach is also based on the Contract-Net protocol. Our contribution was tested by sending 10 to 40 requests which enabled us to measure the latency time. The obtained results are compared with the results obtained using the classic architecture (without specialization). Based on the preliminary results, our proposed architecture shows a significant improvement.

— The second contribution enhanced the previous IaaS. The IaaS proposed in this contribution provides an intelligent storage which ensures load balancing among servers. It also provides a consistent servers monitoring to achieve a high data availability and a rapid data recovery in case of a server failure. To show the performance of our scheme, we compared the latency of an output file request and the complexity of searching for a replicated version of a critical data using the three architectures. Our architecture shows significant improvements in terms of complexity for searching for a file and the latency of servicing an output file request.

— The third contribution addresses the problem of single point failure of the previous IaaS. It also evaluates the performance of ROBUST against LBCS by calculating the latency of critical requests when all servers work properly. Our contribution is implemented using JADE Platform (Java). It has been tested with different number of requests. Compared to LBCS, ROBUST shows significant improvements in terms of latency of critical data recovery from a primary server, and latency when searching for a duplicated critical data in case of a primary server failure.

In chapter three, we propose three contributions that overcome the storage issue in

Wireless Sensors Networks over IoT. We address data availability and latency challenges by designing an efficient storage on IoT-Cloud servers. Next chapter addresses latency and energy by avoiding data retransmissions. Data retransmissions is caused by internal interference, external interference and multi path fading, thus, we discuss three contributions to achieve a reliable wireless sensors communications. Hence, increase the longevity of networks and ensure minimum end to end packet delivery to the end user (e.g., Cloud servers).

Chapter 4

Improvement of Time Slotted Channel Hopping (TSCH)

4.1 Introduction

Most of the IoT applications are purely based on the use of the wireless sensors and actuators technology owing to its numerous, attractive, and critical services in diverse domains. The WSN empowers installations in remote areas and hostile terrain. It is cost efficient, minimizes downtime and improves equipment performance [20]. It is efficiently useful in the ambient environment, human safety and critical products (chemical and pharmaceutical) monitoring, and climate control. Moreover, it cuts across the critical environmental monitoring applications where the real-time monitoring is highly required. The IoT is able to predict the abnormal incidences / disasters so that the appropriate actions will take place automatically before the required deadline. The oil and gas industry applications focus on the human and environmental safety using wireless sensor and actuator networks (WSANs) [20]. The critical data collected by sensors must be delivered in the form of packets as quickly as possible to the base station. The more packets are delivered with minimum end to end delay, the sooner disaster countermeasures are taken, and the safer the environment and the humans are. The real time network high performance is realized when the data captured by a sensor can move through the network and arrive at the base station before the required deadline. Since sensors are small devices, they are equipped with small batteries containing a limited amount of energy. Recharging or changing batteries of thousands of

sensors is a difficult task [20, 103]. Increasing the longevity of the network and minimizing the packet latency are primary goals for such a network. Collisions are one of the main reasons of energy waste [103], they happen when two or more interfering nodes transmit packets at the same time [43].

Diverse communication devices are allowed to operate in the unlicensed band of 2.4 GHz. Those devices equipped with radios that implement the standards IEEE 802.15.1(Bluetooth), IEEE 802.11 (Wi-Fi) and IEEE 802.15.4. The wireless computer peripherals, and microwave ovens are also included. Wi-Fi bandwidth is about 22 MHz. It covers approximately four Wireless Sensor Network (WSN) channels, hence, it deteriorates the QoS of the WSN in the presence of heavy Wi-Fi network traffic [104] (See figure 4.1, [105]). According to empirical studies shown in [104, 106], the packet loss rate due to Wi-Fi interference is approximately 90% of file transfer and 30 % of video streaming. The harsh nature of industrial environment also presents multi-path fading that deteriorates the system performance.

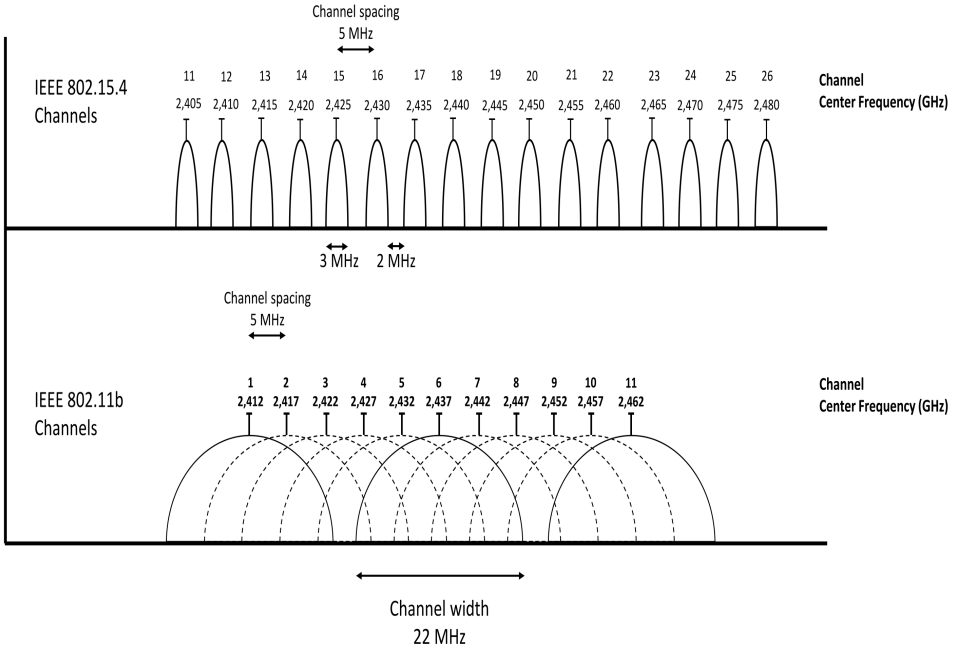

Figure 4.1: IEEE 802.11b and IEEE 802.15.4 spectrum usage.

The IEEE 802.15.4 e [68] standard proposes the Time Slotted Channel Hopping (TSCH) mode designed especially for process automation. It mainly focuses on energy conservation of sensors. It introduces the mechanism of multi channel hopping that

reduces the interference and multi-path fading problems [68]. However, hopping from one frequency to another in a blind manner is not efficient since all frequencies experience different levels of interference [36] [107]. Link quality Estimation (LQE) raised as a fundamental process to select the most stable links for successful communications. This process enhances the network throughput and energy efficiency [97]. TSCH also uses shared links to increase the networks throughput. On the other hand, the shared link nature that allows more than two communicating nodes to transmit packets at the same time may cause collision problem.

Chapter Four is divided into three parts. Each part discusses a contribution that addresses the discussed TSCH challenges.

— We focus in the first part on the TSCH mode by analyzing its backoff algorithm. We propose an improvement of this algorithm in order to avoid internal collisions during the packets retransmissions and to reduce the latency and the energy consumption of the sensors [108].

— The second part presents an accurate link quality estimation process dedicated to TSCH to provide a high performance for the system, to increase the longevity of the network, and to raise the throughput [109].

— In the third part, we propose a new dynamic blacklisting technique dedicated to TSCH mode to improve the reliability of communications. The proposed solution has been simulated using Network simulator 3(NS3). The results show significant improvements in terms of throughput, energy and reliability. This part illustrates the problem statement, explains the link quality estimation process, describes the proposed Enhanced Time Slotted Channel Hoping (E-TSCH)and shows the simulation scenario. The obtained results are also analyzed.

4.2 Fourth Contribution: Time Slotted Channel Hopping with Collision Avoidance

TSCH proposes two retransmission backoff algorithms for both critical and normal packets. The first part of chapter Four analyses the TSCH CSMA-CA backoff retransmission algorithm, explains the problem statement and presents the proposed solution. It also analyses the Priority Channel Access (PCA) backoff algorithm, introduces the

problem statement and the proposed solution [108].

4.2.1 TSCH CSMA-CA backoff retransmission algorithm analysis

In this section we analyze the case of repeated failures of data retransmission. We estimate the packet delivery (delivered from S to R) delay after the maximum allowed retransmission number (*macMaxFrameRetries* times).

Assumptions

— Each time slot lasts 10 ms [14].

— *macMaxFrameRetries* times is equal to 7 [68].

— During these retransmissions a communication on a dedicated link to the destination R will not occur. (to execute the worst case)

— The values of the attributes *macMaxBe* and *macMinBe* are equal to the default values defined in the standard. $macMinBe = 1$, $macMaxBe = 7$ [68].

— The hop time (the time elapsed when switching from one frequency to another) is neglected.

Analysis

Since the variable BE is increased for each successive failure on a shared link until it reaches *macMaxBe* value, the latency time of packet retransmission is calculated as follows:

$$L_i = ((W_i \times TD) \times (M + 1)) + hoptime \times (W_i - 1) \quad (2)$$

L_i is the backoff window of the i^{th} retransmission of a packet. W_i is the random number of the i^{th} retransmission. W_i is picked in $[0, 2^{BE} - 1]$.

TD is the time slot duration. The default value of TD is equal to 10ms.

M is the number of time slots without destination R located between the current shared link (where the collision happens) and the calculated W shared link .

Table 4.1: End to end packet delivery of 7 retransmissions

The number of retransmissions	BE	$[0, 2^{BE} - 1]$	W_1	The end to end time delivery	W_2	The end to end time delivery
1^{st}	1	[0,2^1 - 1] = [0,1]	1	1 × 10 = 10 ms	1	1 × 10 = 10 ms
2^{nd}	2	[0,2^2 - 1] = [0,3]	1	1 × 10 = 10 ms	3	3 × 10 = 30 ms
3^{rd}	3	[0,2^3 - 1] = [0,7]	3	3 × 10 = 30 ms	7	7 × 10 = 70 ms
4^{th}	4	[0,2^4 - 1] = [0,15]	7	7 × 10 = 70 ms	15	15 × 10 = 150 ms
5^{th}	5	[0,2^5 - 1] = [0,31]	15	15 × 10 = 150 ms	31	31 × 10 = 310 ms
6^{th}	6	[0,2^6 - 1] = [0,63]	31	31 × 10 = 310 ms	63	63 × 10 = 630 ms
7^{th}	7	[0,2^7 - 1] = [0,127]	63	630 × 10 = 630 ms	127	127 × 10 = 1270 ms
				1210 ms		**2470 ms**

The end to end packet delivery time (*E2EPD*) in just one hop is calculated as follows:

$$E2EPD = \sum_{i=1}^{n} L_i \qquad (3)$$

$n - 1$ is the failed retransmission attempts.

n is the successful retransmission. n is less than or equal to $macMaxFrameRetries$.

Table 4.1 illustrates the estimated packet delivery latency time after 7 retransmission attempts. The latency time is calculated with $M = 0$.

Since the W shared link is picked randomly from the interval [0, 2^{BE} -1], we propose to assign two values to W (W_1 and W_2) to calculate the latency.

W_1 equals half interval value. W_1 = (2^{BE} -1)/2 .

W_2 equals the maximum interval value (worst case)

The backoff delay of the seventh retransmission attempt is:

1210 ms when $W = W_1$ and 2470 ms when $W = W_2$

Table 4.2 presents the end to end packet delivery time from 1 to 7 retransmission attempts. The end to end packet delivery time is calculated with $W = W_1$ and $M =$ 3, 4, and 6.

Table 4.2: End to end delivery time with $W = W_1$ and M is either 3, 4 or 6

The number of retransmissions	W_1	M=3	*E2EPD*	M =4	*E2EPD*	M=6	*E2EPD*
1^{st}	1	40ms	40ms	50ms	50ms	70ms	70ms
2^{nd}	1	40ms	40+40=80ms	50ms	100ms	70ms	140ms
3^{rd}	3	120ms	120+80=200ms	150ms	250ms	210ms	350ms
4^{th}	7	280ms	280+200=480ms	350ms	600ms	490ms	840ms
5^{th}	15	600ms	600+480=1080 ms	750ms	1350ms	1050 ms	1890ms
6^{th}	31	1240ms	1240+1080= 2320ms	1550ms	2900ms	2170ms	4060ms
7^{th}	63	2520ms	2520+2320=4840ms	3150ms	6050ms	4410ms	8470ms

Table 4.3 presents the estimated end to end packet delivery time from 1 to 7 retransmission attempts. The end to end packet delivery time is calculated with $W = W_2$ and $M = 3$, 4, and 6.

Table 4.3: End to end packet delivery time with $W = W_2$ and M is either 3, 4 or 6

The number of retransmissions	W_2	M=3	*E2EPD*	M =4	*E2EPD*	M=6	*E2EPD*
1^{st}	1	40 ms	40 ms	50ms	50 ms	70ms	70ms
2^{nd}	3	120ms	160ms	150ms	200ms	210ms	280ms
3^{rd}	7	280 ms	440 ms	350ms	550 ms	490ms	770ms
4^{th}	15	600ms	1040ms	750ms	1300 ms	1050 ms	1820ms
5^{th}	31	1240 ms	2280 ms	1550 ms	2850 ms	2170ms	3990ms
6^{th}	63	2520ms	4800 ms	3150ms	6000ms	4410ms	8400 ms
7^{th}	127	5080 ms	9880ms	6350ms	12350ms	8890ms	17290 ms

Figure 4.2 shows the estimated end to end packet delivery time from 1 to 7 retransmission attempts. The end to end packet delivery time is calculated when $W = W_1$ and $W = W_2$ with $M = 3$, 4, 6.

We conclude that the latency increases with the increase of the number of M.

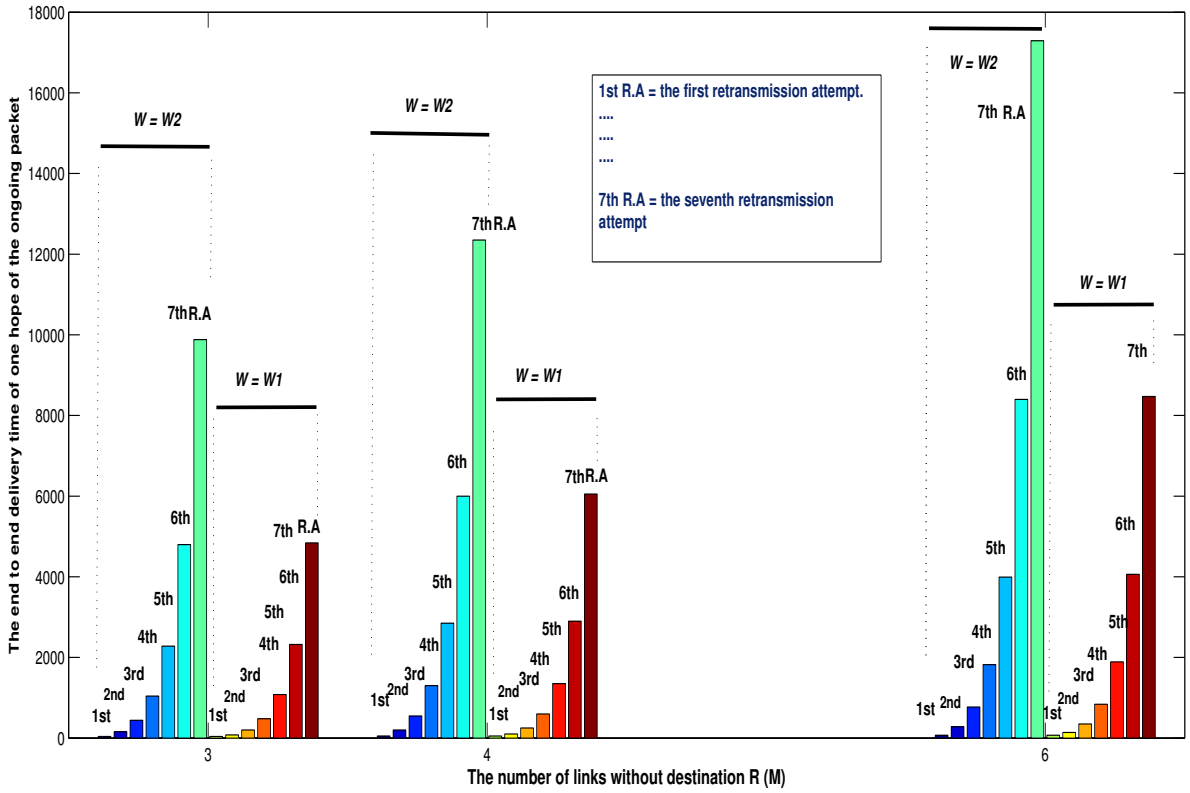

Figure 4.2: End to end packet delivery time when $W = W_1$ and $W = W_2$ with M is either 3, 4 or 6

4.2.1.1 The problem statement

The main reason that leads to the retransmission failure is the random selection of the non appropriate shared link on which the retransmission failed.

Figure 4.3 illustrates the third packet retransmission attempt with $M = 3$ and $W = W_1$.

According to TSCH backoff algorithm, the 3^{rd} packet retransmission attempt is deferred to the shared link number 45.

The retransmission fails on the shared link(with colliding nodes)number 5 during the first retransmission attempt. At the second retransmission attempt, the packet retransmission fails on the shared link (with colliding nodes) number 17 which defers the retransmission onto the shared link number 45.

The packet is delivered to the collision shared link number 17, whilst link number 9 is a collision-free shared link. So, link 9 is a worthy link to transmit data rather than the collision shared link number 17.

The repeated failures of the packet retransmission extended the packet end-to-end

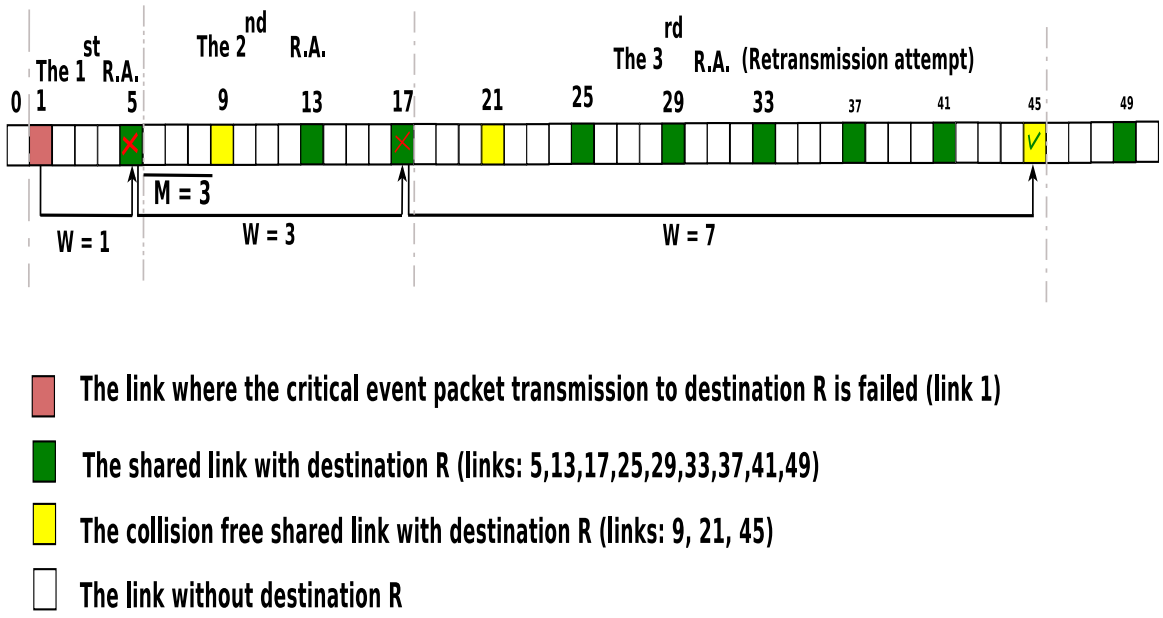

Figure 4.3: Third packet retransmission attempt with TSCH CSMA backoff Algorithm

delay. This point matters in real-time networks where the packets must be delivered to the base station with as little end-to-end delay as possible. Packets in real-time networks must be delivered rapidly, so that the necessary countermeasures are applied before the required deadline. If the appropriate countermeasures do not take place before the deadline, a catastrophic situation may happen and affects the network.

The failed retransmissions hamper the packets delivery of the participating senders scheduled on the same shared link.

The repeated retransmissions influence negatively the network lifetime. Since the packet retransmissions waste dramatically the energy of sensors, which in the long run, decrease the network lifetime.

4.2.1.2 The proposed solution

The idea consists of alleviating the problem of collisions while implementing the TSCH CSMA-CA backoff algorithm. If a transmission fails on a shared link, the packet retransmission should be tightly and carefully scheduled.

In the failed packet transmission, when the W shared link is calculated, the packet retransmission will not be postponed on the W shared link unless this W shared link is a collision-free link.

Assumptions

— Since the TSCH mode is applied for oil and gas industry applications (e.g., checking pipes for gas leak), sensors in such cases are supposed to be static (not mobile).

— For a transmission (S to R), there is an absence of a dedicated link to destination R during the execution of the TSCH CSMA-CA backoff algorithm.

A collision-free link is a link where its participating nodes are out of range from one another, so that they never collide when they communicate. When the W shared link is calculated, the communicating nodes cannot exchange data on this W shared link unless it is a collision-free link. To manage this, we propose to maintain a table that determines the nodes that belong to the same range of each node. The colliding nodes are calculated using received signal strength indication (RSSI).

Figure 4.5 shows an example of some colliding nodes from Figure 4.4

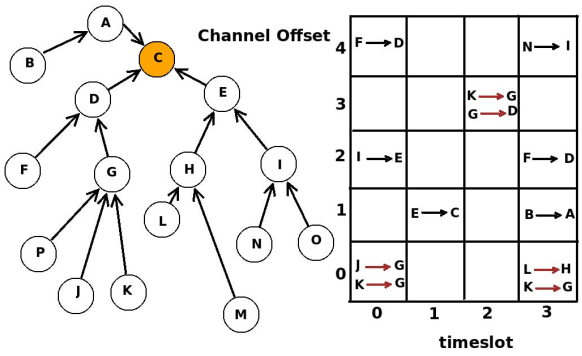

Figure 4.4: Cells scheduling for data collection with a tree network topology

Nodes	The nodes that belong to the same range
G	D , P , J , K , G
K	G , L , K
M	H , N , M
L	H , K , L

Figure 4.5: Colliding nodes

When the sender calculates the W shared link, it checks whether the participating nodes on the W shared link belong to its range or not. The W shared link will not be

scheduled for this retransmission if the receiver receives and sends data on the same link (Figure 4.4, link [3,2]).

The common problems that MAC protocols face are: the hidden node and the exposed node problem. The TSCH with Correct Collision Avoidance CSMA-CA (TSCH-CCA CSMA) backoff algorithm deals with these problems as follows:

The exposed node problem (Figure 4.4, link[0,3]) occurs when the sender falsely concludes that the transmission may not happen on the W shared link. Because the participating transmitter nodes on the W shared link belong to its range, even though if both transmissions are on the same link they certainly would succeed because the receivers belong to different ranges (Figure 4.6). In such a case, when the sender checks that the receivers on the W shared link do not belong to its receiver's range, it will normally schedule the retransmission on this W shared link.

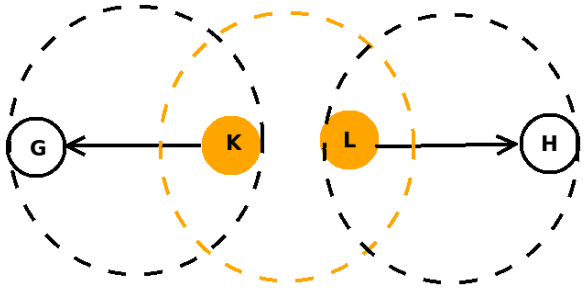

Figure 4.6: Exposed node problem

The hidden node problem occurs when the senders of two transmissions belong to different ranges, while the receiver belongs to these ranges (Figure 4.7). To avoid such a problem, the sender verifies whether the communicating receiver nodes on W shared link belong to its receiver's range. If yes, the sender does not retransmit on that link.

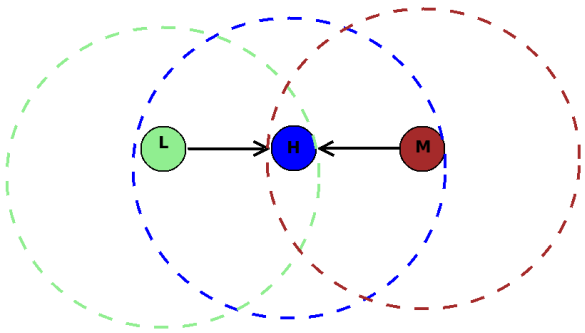

Figure 4.7: Hidden node

In order to solve this problem, we propose a different implementation of the TSCH

CSMA backoff algorithm. The proposed algorithm called ***TSCH-CCA CSMA-CA backoff algorithm*** .

Algorithm 16 TSCH-CCA CSMA-CA backoff Algorithm

```
1: BE := macMinBE;
2: NB:= 0;
3: Generate a random number W in [0 , 2^BE-1].
4: Do
5: if (W is a collision-free shared link) then
6:    Delay for W shared link
7:    Retransmission on W shared link
8:    if (the Acknowledegement is not received) then
9:       NB:= NB + 1;
10:      BE:= Min (BE +1 , macMaxBE);
11:      Generate a random number W in [0 , 2^BE-1];
12:   end if
13: else
14:   W:= W+1
15:   W: next encountered shared link to destination R;
16: end if
17: WHILE(Acknowledegement is received) Or (NB > macMaxFrameRetries)
18: if The retransmission is acknowledged then
19:   BE:= macMinBE;
20: else
21:   Drop the packet
22: end if
23: END
```

In TSCH-CCA CSMA-CA backoff Algorithm (**Algorithm 2**), the backoff period is calculated according to the shared links' state. The shared link may be a collision-free link or a collision link.

This algorithm avoids retransmitting the packets on shared links with colliding nodes. On the other hand, it seeks to establish the retransmissions on the nearest collision-free shared link.

Before scheduling the retransmission, the sender checks first whether the selected shared link is a collision-free or not. If the shared link is a collision-free, it schedules the retransmission on it; otherwise, it defers the retransmission to nearest incoming collision-free shared link to destination R.

The TSCH-CCA CSMA-CA backoff Algorithm is applied only when the collision

Table 4.4: Estimation of the packet delivery time using TSCH-CCA CSMA-CA backoff

The number of retransmission attempts	M=3	M =4	M=6
The 1^{st} encountered collision shared link (link number 5)	40 ms	50 ms	70 ms
The 1^{st} encountered collision-free shared link (link number 9)	40 ms	50 ms	70 ms
The end to end packet delivery	80 ms	100 ms	140 ms

Table 4.5: Estimation of the delivery time gain when using TSCH-CCA CSMA-CA backoff algorithm

The number of retransmissions	M=3	M =4	M=6
3^{rd}	440-80 = 360 ms	550- 100 = 450 ms	770-140 = 630 ms
4^{th}	1040-80= 960 ms	1300-100 = 1200 ms	1820-140 = 1680 ms
5^{th}	2280-80= 2200 ms	2850-100= 2750 ms	3990-140 = 3850 ms
6^{th}	4800-80= 4720 ms	6000-100= 5900 ms	8400-140 = 8260 ms
7^{th}	9880-80= 9800ms	12350-100= 12250 ms	17290-140 = 17150 ms

happens on a shared link.

Table 4.4 illustrates the estimated delivery latency of the packet retransmission illustrated in Figure 4.3 when using the TSCH-CCA CSMA-CA backoff algorithm.

According to the example illustrated in Figure 4.3, when we apply the TSCH-CCA CSMA-CA backoff algorithm, we can see that the packet is successfully delivered in the first retransmission attempt after 80 ms (when $M = 3$). When we apply the TSCH CSMA-CA backoff algorithm, if the 3^{rd} retransmission attempt is scheduled randomly on a collision-free shared link, the packet can be successfully delivered from the 3^{rd} till the 7^{th} retransmission attempts (when $M = 3$).

Table 4.5 illustrates the gain in terms of packet delivery time.

Table 4.6: Estimation of the residual energy of a sensor using TSCH-CCA CSMA-CA and TSCH CSMA-CA backoff algorithms

The number of retransmissions	The residual energy (TSCH CSMA-CA)	The residual energy (TSCH-CCA CSMA-CA)
The failed transmission	2000 - 80 = 1920 mw	2000 - 80 = 1920 mw
The 1^{st} retransmission	1920 - 80 = 1840 mw	1920 mw
The 2^{nd} retransmission	1840 - 80 = 1760 mw	1920 mw
The 3^{rd} retransmission	1760 - 80 = 1680 mw	1920 mw
The 4^{th} retransmission	1680 - 80 = 1600 mw	1920 mw
The 5^{th} retransmission	1600 - 80 = 1520 mw	1920 mw
The 6^{th} retransmission	1520 - 80 = 1440 mw	1920 mw
The 7^{th}retransmissions	1440 - 80 = 1360 mw	1920 - 80 = 1840 mw

Because of the tight schedule of the retransmission assignments, when using TCH-CCA CSMA-CA backoff algorithm the packet is almost delivered successfully at the first retransmission attempt.

Table 4.6 presents the residual energy of a sensor using TSCH-CCA CSMA-CA and TSCH CSMA-CA backoff algorithms. Assuming that the sensor initially has 2000mw.

Figure 4.8 presents the residual energy of a sensor using TSCH-CCA CSMA-CA and TSCH CSMA-CA backoff algorithms after successfully retransmitting one packet. It shows the worst case where the first 6 shared links which are randomly calculated are collision shared links (while the shared link calculated for the seven retransmission attempt is a collision-free). Note that in the TSCH CSMA-CA algorithm, a packet has failed to be transmitted six times. These successive failures are due to the random selection of the shared link. Since the TSCH-CCA CSMA-CA algorithm checks first the nature of the link (Collision-free or not) before retransmitting in all these pre-calculated shared links, it prevents the sender from draining energy owing to successive failed retransmissions. Note that the TSCH-CCA CSMA-CA conserves the energy six times (480 mw) more than the TSCH CSMA-CA backoff algorithm by successfully transmitting just one packet. In the worst case, this time can be exploited to transmit three other packets using TSCH-CCA CSMA-CA.

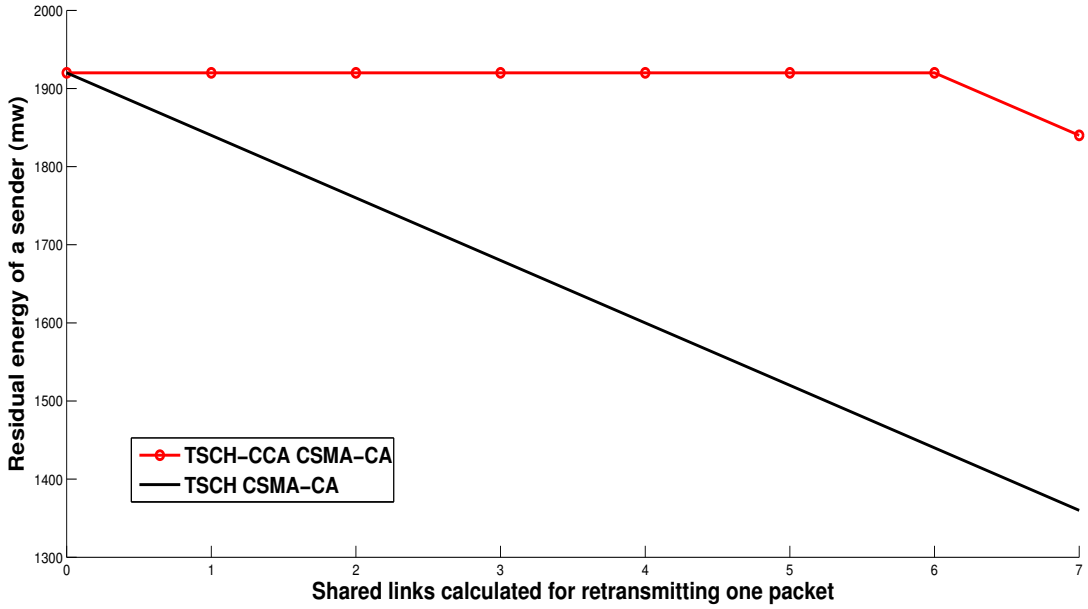

Figure 4.8: Estimated energy drain using TSCH-CCA CSMA-CA and TSCH CSMA-CA backoff algorithm

The energy spent (ES in mw) is calculated as follows:

$$ES = ETF + (NBT \times 80) \qquad (4)$$

ETF is the energy spent during the packet transmission failure. $ETF = 80$ mw.

NBT is the number of retransmissions.

$1 \leq NBT \leq 7$

Since the TSCH-CCA CSMA-backoff Algorithm assigns the retransmission to the appropriate collision-free node, the first retransmission will have a high probability to be successfully transmitted. In such a case, the NBT variable is equal to 1, so the energy spent in this case is equal to 160 mw.

Theorem 1

The failed transmission will be successfully delivered at the first retransmission attempt despite the absence of a dedicated link.

Proof

When a $packet_i$ fails to be transmitted, the sender S_i calculates a random backoff $W \in [1, 2^{BE}\text{-}1]$ to retransmit the packet. If we suppose that the W shared link contains

colliding nodes that host packets to deliver (such as, hidden nodes) on the W shared link, the $packet_i$ inevitably fails again to be retransmitted since the sender S_i did not check the state of the shared link.

The TSCH-CCA CSMA-CA backoff algorithm provides correct collision avoidance since it checks the participating nodes of the shared link before retransmitting the packet. If a sender S_i checks the shared link and finds that at least one receiver belongs to its receiver's influential range, it prevents the retransmission; otherwise it allows it. Thus, it provides collision avoidance even in the presence of hidden nodes.

Theorem 2

TSCH-CCA CSMA-CA backoff algorithm extends the network lifetime.

Proof

When the retransmission fails in $(W_i, W_{i+1}, .W_k)$ shared links (where k is less or equal to the allowed number of packet retransmission (k =7)), the sender automatically drains about ($k\times$ 80) mw of energy in transmitting just one packet. Since a $node_i$ transmits N packets during the network's life cycle, the $node_i$ drains at most $N\times$ ($k\times$ 80) mw of energy. If the network deploys M nodes, the total energy drain of all sensors deployed in such a network is at most equal to:

$$(M \times (N \times (k \times 80))mw \qquad (5).$$

TSCH-CCA CSMA-CA reduces the energy consumption k-1 times since it ensures the packet delivery success from the first retransmission attempt.

Theorem 3

TSCH-CCA CSMA-CA Backoff algorithm avoids the network congestion.

Proof

If a sender S_i retransmits its packet on the calculated W collision shared link without checking the state of this W shared link, a collision will likely happen. In such a case, the sender S_i hampers the packet delivery of the participating senders scheduled on the same shared link. This causes the senders to execute the TSCH algorithm to retransmit their packets. The repeated retransmissions result in the network congestion.

TSCH-CCA CSMA-CA backoff algorithm does not cause the network congestion since it selects the appropriate shared link on which the packets will be delivered successfully.

Theorem 4

TSCH-CCA CSMA-CA backoff algorithm minimizes the latency of the packet delivery.

Proof

In each packet retransmission R_i attempts the backoff window increases to mitigate the collision problem. Assuming that each retransmission R_i takes Δ_{ti} to be started, and if the packet retransmission fails to be transmitted K consecutive times, the end to end delay of the successful packet delivery in just one hop is equal to: Δ_{tT} ; where $\Delta_{tT} = \sum_{i=1}^{k} \Delta_{ti}$; $1 \leq K \leq 7$

If we suppose that H is the number of hops between sender S_i and the final destination D_i, so the packet latency will be at most equal to: $\Delta_{ti} \times H$ (ms). Clearly the increase of the backoff window influences negatively on the packet latency. This problem occurs due to the random selection of the W shared link.

TSCH-CCA CSMA-CA backoff algorithm does not increase the backoff window in a random manner. The backoff window is increased according to the absence of a collision-free shared link.

The backoff time may be too short when the collision-free shared link is encountered, as it may be long, but with correct collision avoidance.

In conclusion, postponing the retransmission on a collision-free shared link with high probability of packet delivery success does not cost too much (in terms of delay, energy, and network congestion) compared to postponing the retransmission to the non appropriate W shared link (picked randomly).

4.2.2 Priority Channel Access Backoff Algorithm Analysis

This section analyses the delivery time of the PCA backoff Algorithm.

The variable TB and M take these values:

TB= 3, 7 and 11.

M = 3,4,6 and 8.

The variable M is the number of links without destination R located between the link to the destination R (where the transmission failed) and the nearest shared link to destination R (Figure 4.9).

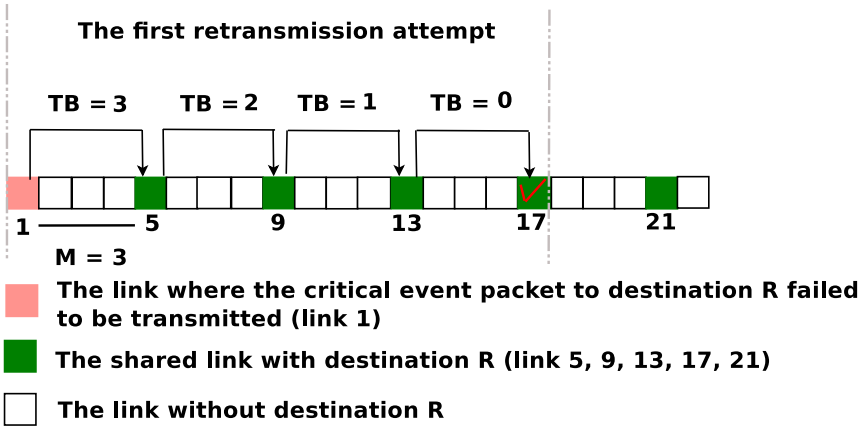

Figure 4.9: PCA backoff retransmission mechanism

Table 4.7 shows the estimation of the first end to end packet transit time (in ms) taking into account the parameters cited above.

The end to end packet delivery of the first retransmission of a critical event packet is calculated as follows :

$$TD \times (TB + 1) \times (M + 1) \qquad (6)$$

TD is the time slot duration. The default value is equal to 10 ms.

Table 4.7: End to end backoff time of a critical event packet delivery using PCA backoff algorithm

TB	M =3	M =4	M =6	M =8
3	160 ms	200 ms	280 ms	360 ms
7	320 ms	400 ms	560 ms	720 ms
11	480 ms	600 ms	840 ms	1080 ms

4.2.2.1 The main problem

When the retransmission of a critical event packet is referred for TB+1 shared links, and when the TB+1 shared link contains colliding nodes, the critical event packet retransmission will probably fail. In this case, the packet will be retransmitted until it is delivered successfully (at most NB times) (See Figure 4.10).

Figure 4.10 shows the backoff period of the second retransmission attempt.

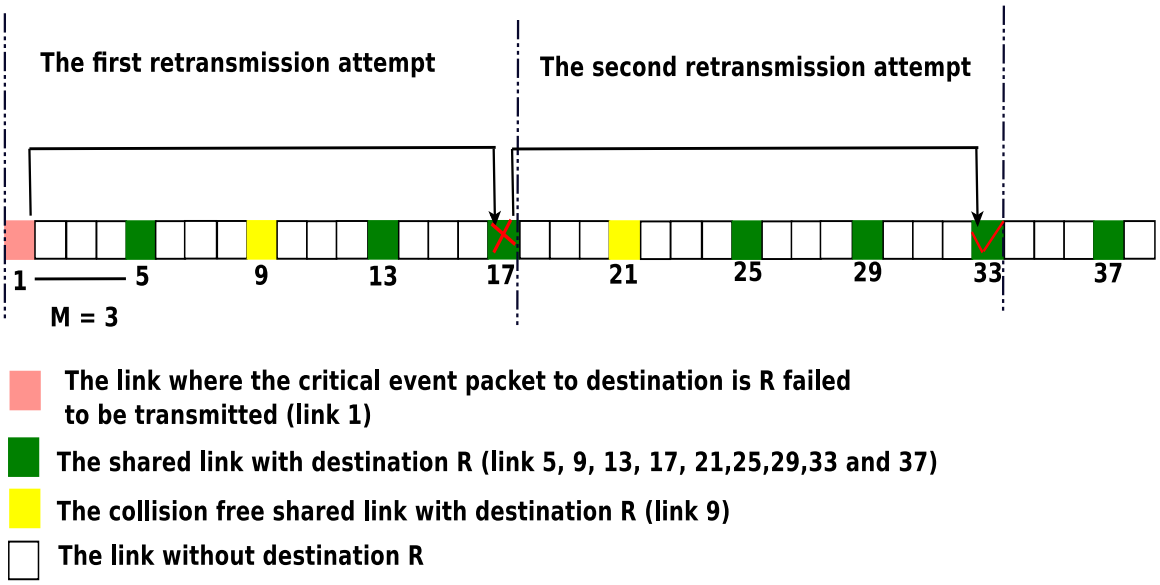

Figure 4.10: Backoff delay after the 2^{nd} retransmission attempt

When the retransmission fails at the first retransmission attempt, it will be postponed for another backoff period to be retransmitted. If the packet is not successfully delivered, the transmission will be allowed to be repeated $macMaxFrameRetries$ times as long as the elapsed time does not exceed the $MacCritMsgDelayTol$ of the current packet.

Tables 4.8, 4.9 and 4.10 and, Figures 4.11, 4.12 and 4.13 show the analyses of the backoff period of the packet delivery from the 1^{st} retransmission attempt until the 7^{th} retransmission attempt.

The backoff delay of each retransmission is calculated as follows:

$TD \times (TB + 1) \times (M + 1) \times RT$ (7)

RT is the number of retransmissions.

Postponing the packet delivery influences negatively the critical event packet lifetime.

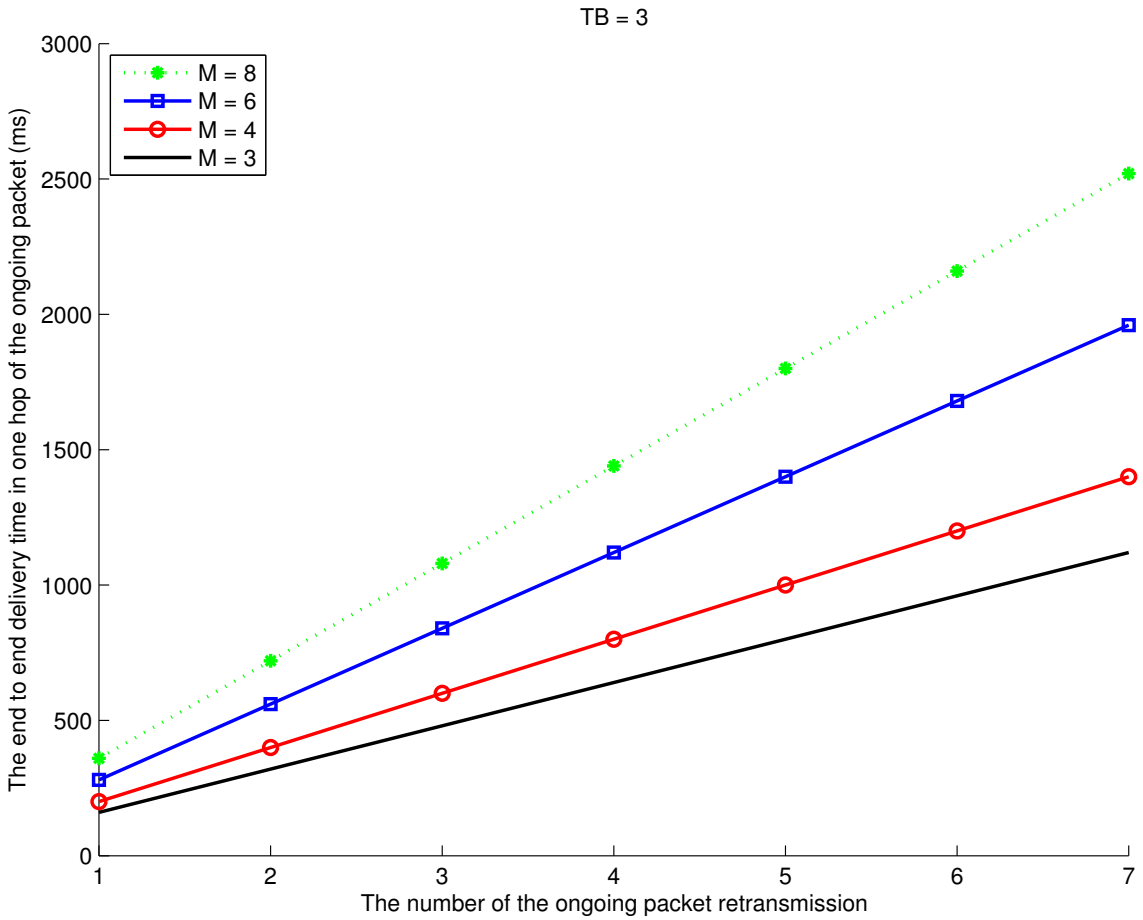

Figure 4.11: End to end packet delivery latency until the 7^{th} retransmission attempt (TB=3)

4.2.2.2 The problem statement

The main problem that led to the first retransmission failure is the selection of the non appropriate link (TB+1 link).

In the case of the absence of a dedicated link, the algorithm retransmits the packet on the TB+1 shared link without taking into consideration the state of that shared link. When the TB+1 is not the appropriate shared link , collision will inevitably happen. Thus, the backoff period of the retransmission increases as illustrated in (Figure 4.10).

4.2.2.3 The proposed solution

4.2.2.3.1 The main idea: In the PCA backoff algorithm, during the backoff period, the transmitter may encounter collision-free shared links less than TB+1 (Figure 4.14, link 9) while the TB+1 shared link (on which the retransmission is postponed) contains colliding nodes.

Table 4.8: End to end packet delivery latency analysis until the 7^{th} retransmission attempt (TB= 3)

The number of retransmissions	$M = 3$	$M = 4$	$M = 6$	M =8
The 1^{st} retransmission	160 ms	200 ms	280 ms	360 ms
The 2^{nd} retransmission	320 ms	400 ms	560 ms	720 ms
The 3^{rd} retransmission	480 ms	600 ms	840 ms	1080 ms
The $4^{th}retransmission$	640 ms	800 ms	1120 ms	1440 ms
The 5^{th} retransmission	800 ms	1000 ms	1400 ms	1800 ms
The 6^{th}retransmissions	960 ms	1200 ms	1680 ms	2160 ms
The 7^{th} retransmission	1120 ms	1400 ms	1960 ms	2520 ms

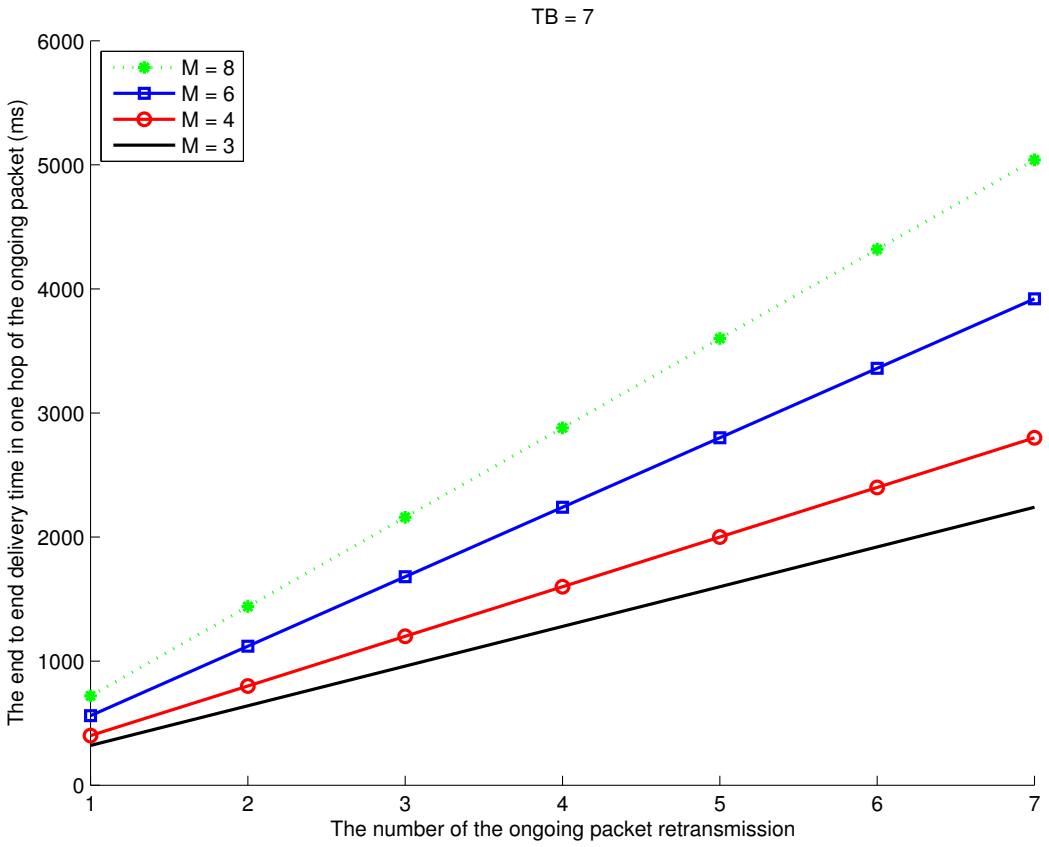

Figure 4.12: End to end packet delivery until the 7^{th} retransmission attempt (TB= 7)

Table 4.9: End to end packet delivery analysis until the 7^{th} retransmission attempt (TB= 7)

The number of retransmissions	$M = 3$	$M = 4$	$M = 6$	M =8
The 1^{st} retransmission	320 ms	400 ms	560 ms	720 ms
The 2^{nd} retransmission	640 ms	800 ms	1120 ms	1440 ms
The 3^{rd} retransmission	960 ms	1200 ms	1680 ms	2160 ms
The 4^{th} retransmission	1280 ms	1600 ms	2240 ms	2880 ms
The 5^{th} retransmission	1600 ms	2000 ms	2800 ms	3600 ms
The 6^{th} retransmissions	1920 ms	2400 ms	3360 ms	4320 ms
The 7^{th} retransmission	2240 ms	2800 ms	3920 ms	5040 ms

Table 4.10: End to end packet delivery latency analysis until the 7^{th} retransmission attempt (TB= 11)

The number of retransmissions	$M = 3$	$M = 4$	$M = 6$	M =8
The 1^{st} retransmission	480 ms	600 ms	840 ms	1080 ms
The 2^{nd} retransmission	960 ms	1200 ms	1680 ms	2160 ms
The 3^{rd} retransmission	1440 ms	1800 ms	2520 ms	3240 ms
The 4^{th} retransmission	1920 ms	2400 ms	3360 ms	4320 ms
The 5^{th} retransmission	2400 ms	3000 ms	4200 ms	5400 ms
The 6^{th} retransmissions	2880 ms	3600 ms	5040 ms	6480 ms
The $7^{th} retransmission$	3360 ms	4200 ms	5880 ms	7560 ms

The Enhanced PCA backoff algorithm aims to minimize the backoff period and tries to deliver the packet successfully at the first retransmission attempt. The main idea consists of increasing the backoff period and analysing each shared link to destination

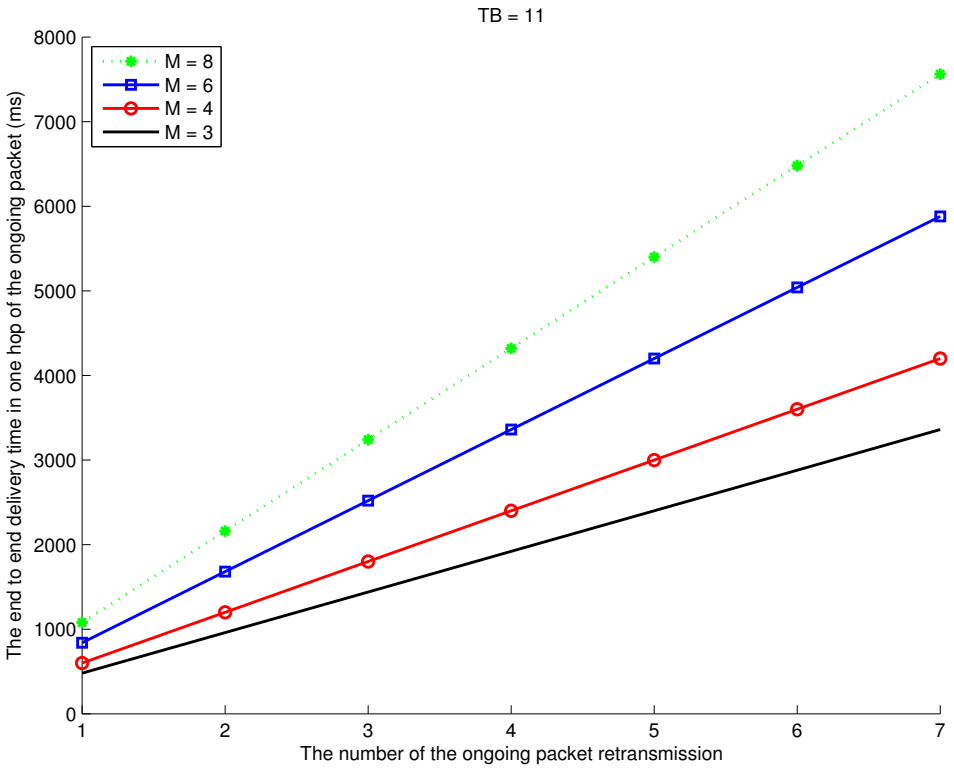

Figure 4.13: End to end packet delivery latency until the 7^{th} retransmission attempt (TB= 11)

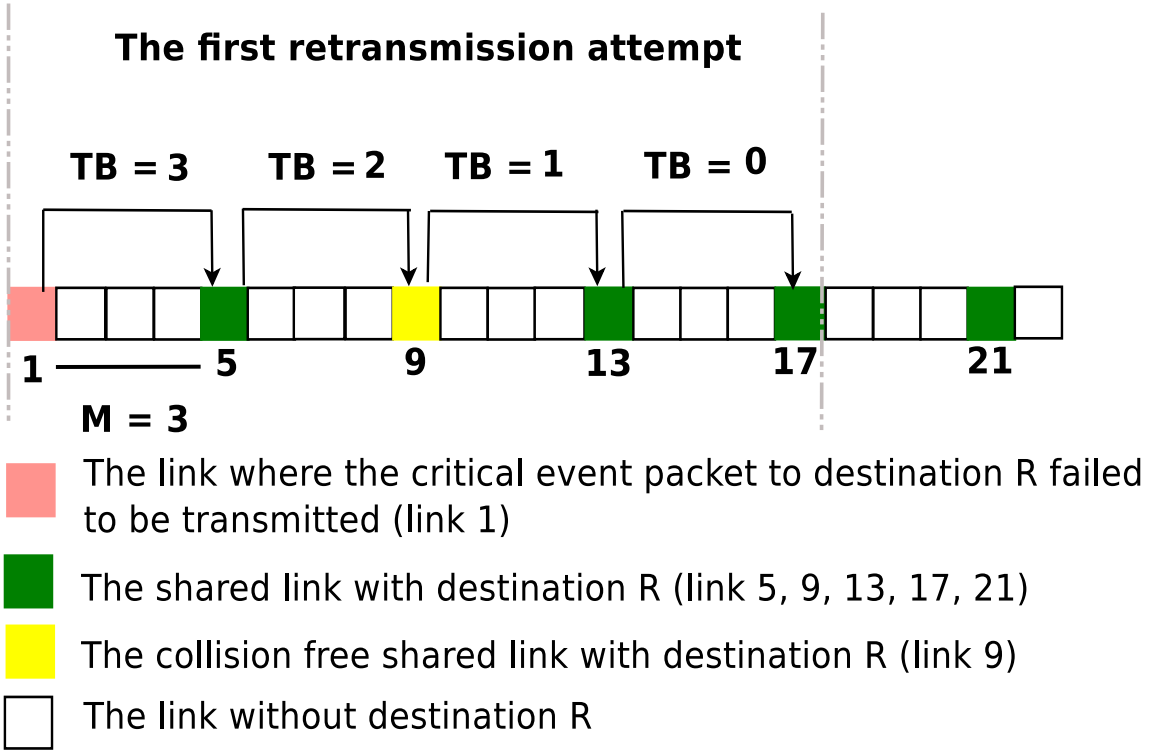

Figure 4.14: First critical packet retransmission attempt without shared link analysis.

R (less than TB+1) encountered. If the sender encounters collision-free shared link

(Less than TB+1) to destination R, the transmitter retransmits the packet on this shared link (See figure 4.16).

Figure 4.15 illustrates the first critical event packet retransmission attempt with shared link to destination R analysis.

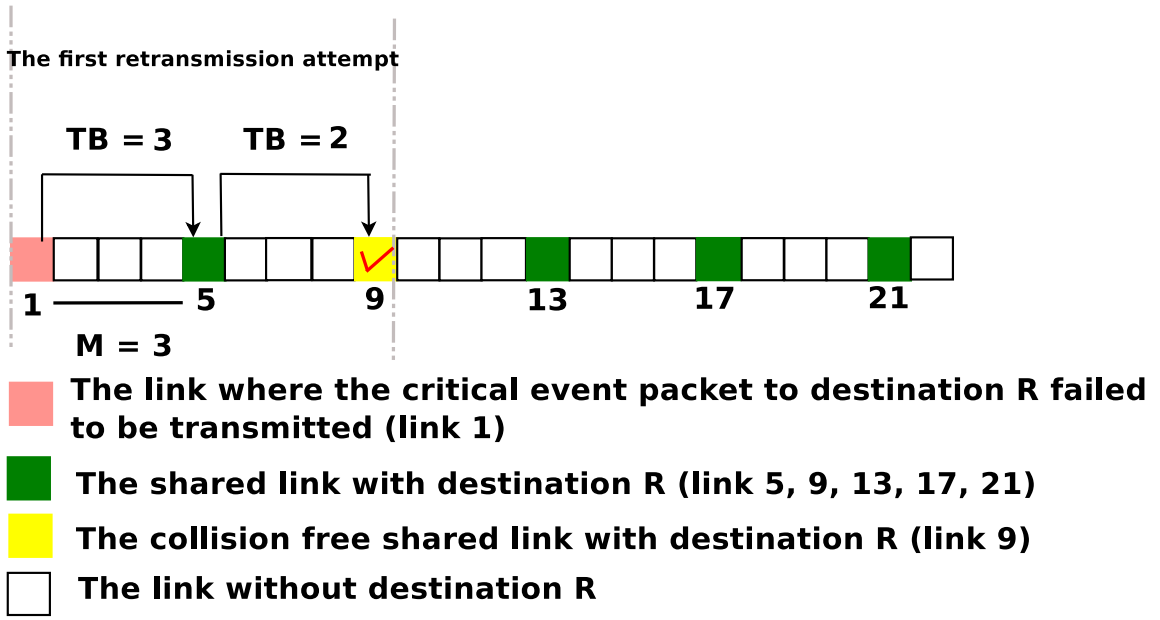

Figure 4.15: First critical packet retransmission attempt with shared link analysis.

4.2.2.3.2 The details: In the Enhanced PCA Algorithm the collision-free shared link is a link.

— without colliding nodes and,

— with high trust.

Since the creation of the network, all transmissions scheduled on that link must be successfully done. In other words, a high trust link is a link that has never been subjected to collisions before. We propose an algorithm called Trust-estimation **Algorithm 17** that calculates the trust of all shared links since the starting of the network. The Trust-estimation [The number of channels * The slotframe size] is a matrix. The rows of this matrix represent the Channels offset. The columns represent the number of time slots.

Table 4.11 calculates the backoff delay of the critical event packet retransmission algorithm according to the example illustrated in Figure 4.15.

Figure 4.17 shows the end to end critical packet delivery according to the number of shared links without desitination R and the variable TB.

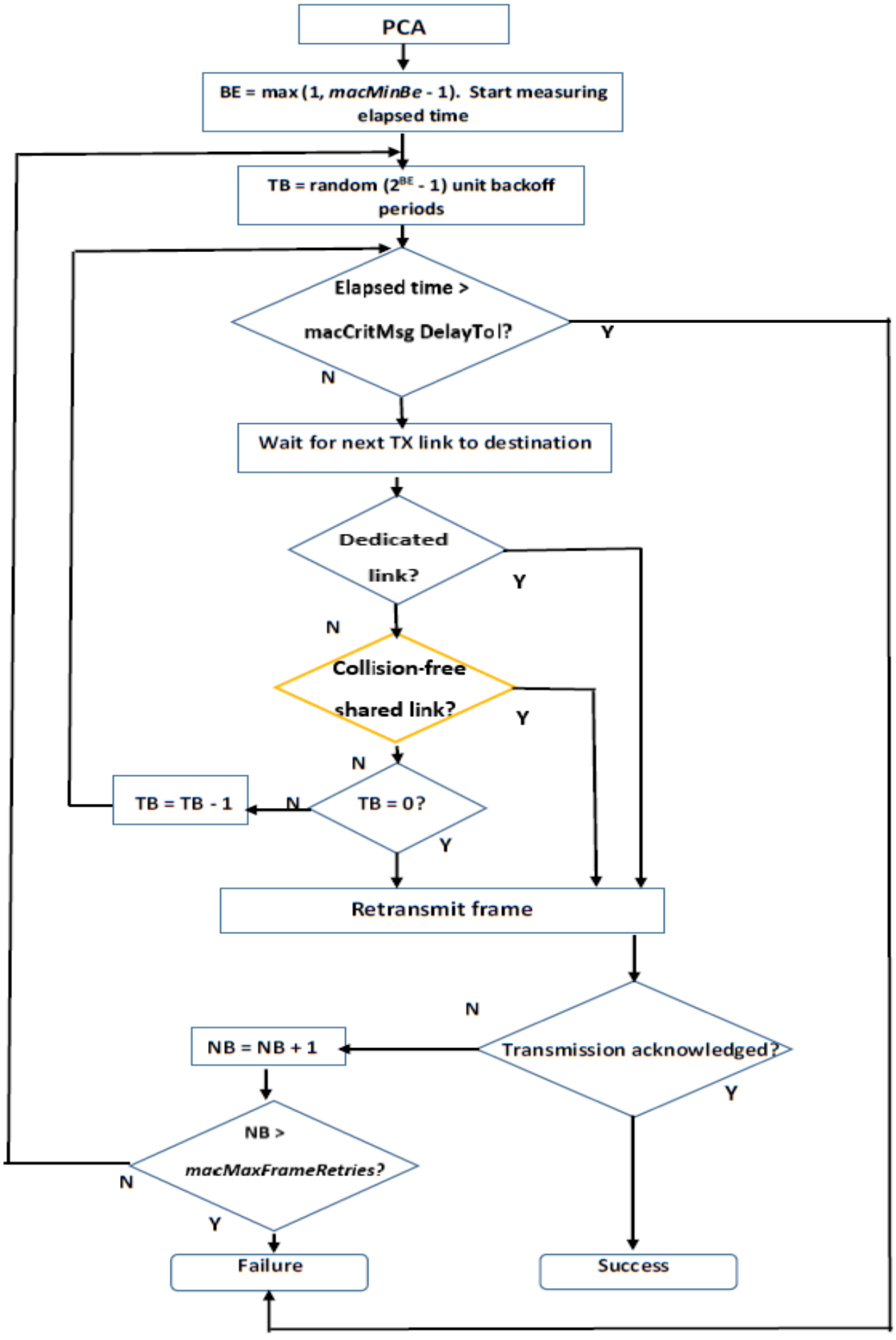

Figure 4.16: Enhanced PCA retransmission algorithm.

Algorithm 17 Trust-estimation ALGORITHM

BEGIN
Variables:
% The matrix that hold the state of each link: Integer
Collision_estimation [][]:= 0; Integer $Ch \in [11, 26]$;
Integer TS:= 0;
% ASN is the number of time slot since the start of the network
Integer ASN:=0;
Constants
% size: is the size of the slotframe (The number of time slots in each slotframe);
Integer *Size*;

if $(ASN \geq size)$ **then**

  if (Collision) **then**
    Collision_estimation $[Ch][ASN]$ ++ ;
  end if
else
  if (Collision) **then**
    $TS := ASN$ mode *size* ;
    Collision_estimation $[Ch][TS]$ ++ ;
  end if
end if
END

Table 4.11: End to end packet delivery when using the enhanced PCA retransmission algorithm.

The number of retransmissions	$M = 3$	$M = 4$	$M = 6$	M =8
The 1^{st} outcoming collision shared link	40 ms	50 ms	70ms	90 ms
The 1^{st} outcoming collision-free shared link	40 ms	50 ms	70ms	90 ms
The end to end packet delivery	80 ms	100 ms	140 ms	180 ms

From Figure 4.17 we remark that the latency rises with the increase of the number of shared links without destination R and the number of TB.

Figure 4.18 shows the end to end critical event packet delivery from one to seven retransmission attempts when using PCA backoff and Enhanced PCA backoff algorithms.

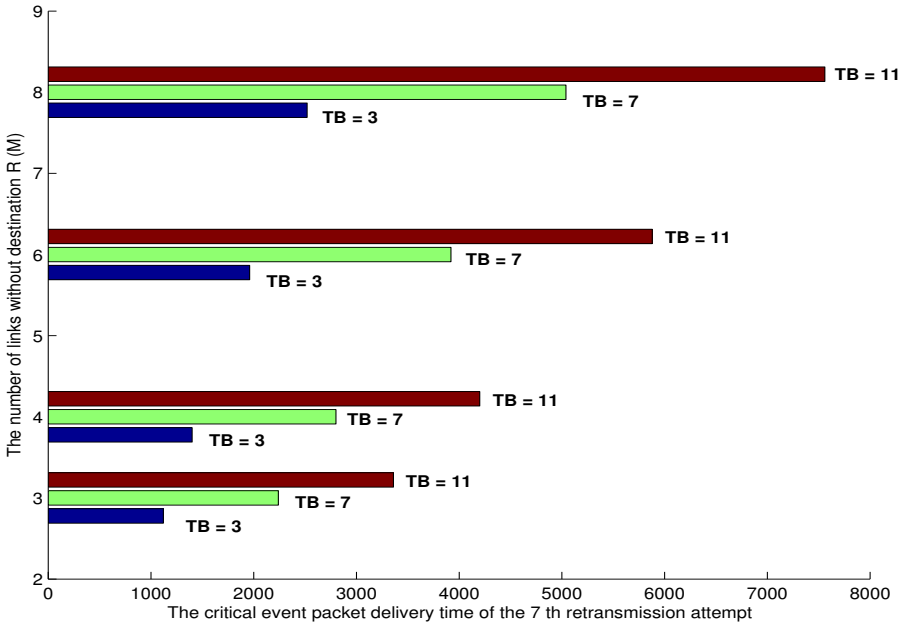

Figure 4.17: Critical event packet delivery according to TB and M

Theorem 1

The Enhanced Priority channel access algorithm (E-PCA) ensures the critical event packet delivery before the *macCrtiMsgDelayTol* time elapses.

Proof

When $TB = 11$, the packet successfully transmitted after 3360 ms (in just one hop). Assuming that H is the number of hops between the sender and the final destination where $(30 \geq H \geq 5)$. The packet will be delivered successfully after at most $H \times 3360$ ms. In such a case, the packet risks to be dropped before it reaches the final destination since the default lifetime of a Critical event packet is equal to 15000 ms. In the case where $M = 3$, the packet risks to be dropped at the fourth hop.

The E-PCA aims to transmit the packet successfully at the first attempt to ensure the packet is delivered to the final destination. Using the E-PCA the packet will be delivered successfully in one hop after 480ms. In such a case, the packet lifetime is preserved for 32 hops.

Theorem 2

The Enhanced Priority channel access (E-PCA) algorithm avoids collision.

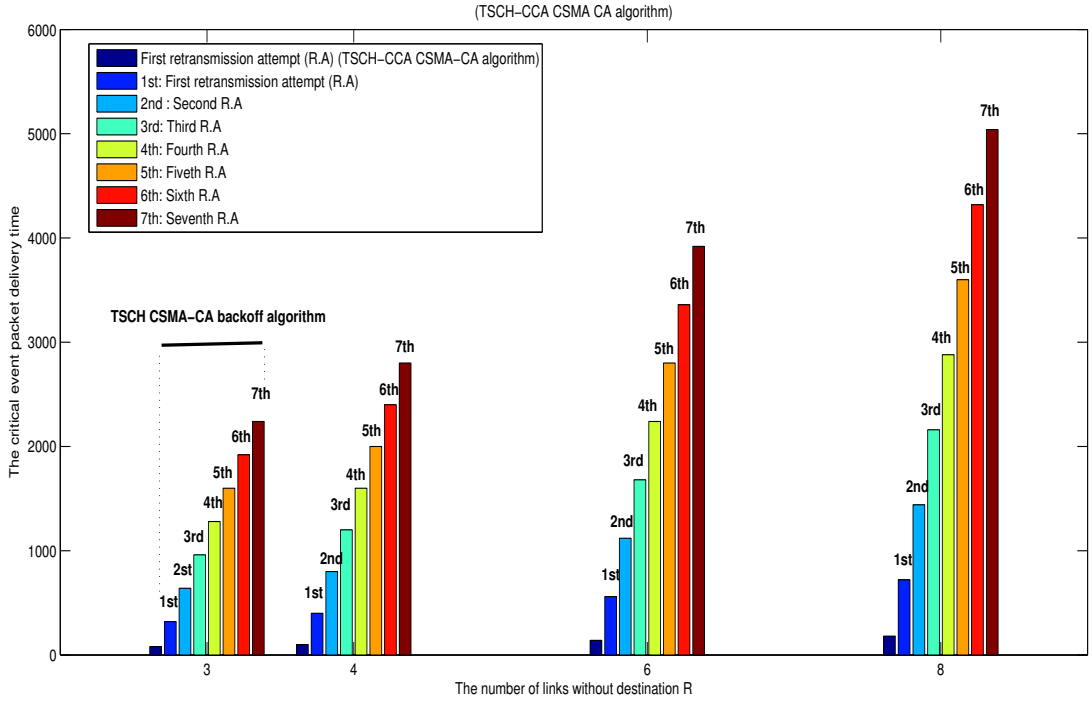

Figure 4.18: End to end packet delivery when using PCA backoff and Enhanced PCA backoff algorithms.

Proof

When the critical event packet transmission fails the packet will be randomly postponed for TB shared links. The algorithm transmits the critical packet without checking the state of this shared link. If $TB+1$ shared link contains hidden nodes, a collision inevitably happens. When the retransmission fails, the sender postpones the packet to another $TB+1$ shared link.

The E-PCA algorithm postpones the packet to the Collision-free shared link. It checks the participating nodes and the trust of each link before retransmitting the packet. If the shared link to destination R is a link without colliding nodes, and if it has a high trust, then the sensor proceeds with the retransmission; otherwise it postpones the retransmission to the nearest collision-free shared link.

Theorem 3

The Enhanced Priority Channel Access (E-PCA) algorithm extends the network lifetime and prevents the network congestion.

Proof

If a packet is randomly retransmitted to $TB+1$ shared link without checking its state, this random selection risks the packet retransmission failure. Since the sensors consume a lot of energy while transmitting a packet, in the long run, the repeated retransmission failures dramatically decrease the network lifetime. The packet retransmission failure on a shared link obstructs the packet transmissions scheduled on the same shared link.

The E-PCA prolongs the network lifetime since it minimizes the energy consumption of sensors by avoiding collision. It also avoids the network congestion since it selects the collision-free shared link on which it retransmits the packet.

4.3 Fifth Contribution: External Interference free Channel Access Strategy dedicated to TSCH

In the previous part, we treat internal collisions. The second part deals with external interference and multi path fading. It presents an accurate link quality estimation process dedicated to TSCH to provide a high performance for the system, to increase the longevity of the network, and to raise the throughput. The rest of the second part illustrates the problem statement, describes the proposed solution, shows the simulation scenario and analyses the obtained results [109].

4.3.1 The problem statement

Blacklisting the channels that experience external interference and multi-path fading enhances the network performance. On the other hand, the internal interference along with the external interference and the harsh nature of the environment makes the link quality diagnostic less accurate. When a packet fails to be transmitted due to a multi-path fading or external interference, this indicates that the channel may be subjected to be blacklisted; otherwise the channel quality is considered good since the packet failure is due to colliding nodes on a shared link. Estimating a channel quality using only the number of packet failures without taking into account the interference sources may not give an accurate channels' quality diagnosis. Over time, this

inaccuracy deteriorates the system performance. Thus, we propose a new technique that provides more accuracy in channels quality diagnosis. The technique is based on identifying the packet transmission failure source before judging the link's quality.

Figure 4.19 shows the possible cells scheduling for data collection in a simple network that has a tree topology.

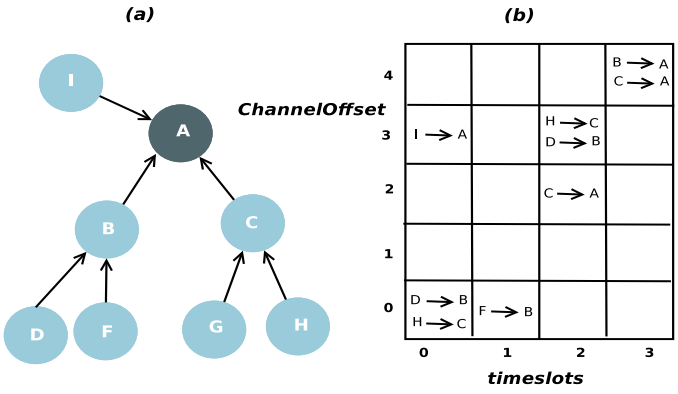

Figure 4.19: The possible cells scheduling in a tree topology.

Link [3,0] is a dedicated link. Link[0,0] is a shared link. Unlike the dedicated link, the shared link intentionally holds more than two communicating nodes.

4.3.2 The proposed solution

In this section we diagnose the source of a packet transmission failure. We also estimate the channels' link quality and blacklist the undesirable channels affected just by an external interference or a multi-path fading.

Assumptions

— Nodes are not mobile (stationary).

— Network is characterized by link layer asymmetry (A node can not transmit packets on a blacklisted link, but it can receive packets on it).

— Only two events are used for estimating a channel quality (external interference and multi-path fading).

4.3.2.1 Packet transmission failure provoker diagnosis

We identify three reasons that provoke a packet transmission failure in a $link_i$. External interference, multi-path fading, and internal interference are the main ones.

Because the TSCH mode allows links to be intentionally shared links, we diagnose the source of a packet transmission failure based on link_i type. We classify each link_i in the TSCH schedule into two main types: collision-free link and collision link. If link_i is a collision-free, then we conclude that the transmission failure provoker is either an external interference or a multi path fading; otherwise the failed transmission provoker may be an internal interference.

To determine whether a link_i is a collision-free or a collision link, we analyze the pre-scheduling nodes in each link. If a link_i is a dedicated link (no collision based link can happen), then link_i is a collision-free. A shared link can also be either a collision-free or collision shared link. A collision-free shared link is a link where its participating nodes are not colliding nodes. While the collision shared link is a link where its participating nodes are colliding nodes (see Algorithm 18).

Algorithm 18 A link nature diagnosis

```
BEGIN
if (link_i is a shared link) then
  if link_i contains colliding nodes then
    link_i-state = 'collision';
  else
    link_i-state = 'collision-free';
  end if
else
  link_i-state = 'collision-free';
end if
END
```

Collision-free shared link that contains non colliding nodes is a link where its participating nodes are out of the range of each other. When two or more senders transmit packets at the same time, these packets will never collide with each other. Collision shared link's participating nodes packets can collide with each other when they are hidden nodes (see Figure 4.20) (see link [4.3] in Figure 4.19.b)

To enable sensors scheduled in a shared link to distinguish between these two kinds of link, each sender_i creates a table that holds its receivers' neighbors (by using received signal strength indicator (RSSI)). Table 4.12 shows the nodes' neighbors of each node of the tree topology illustrated in Figure 4.19.(a).

If sender_i finds that shared link_i contains hidden nodes, then it stores the type of this link in a variable called *link_i-state* as a collision link; otherwise it stores it as a

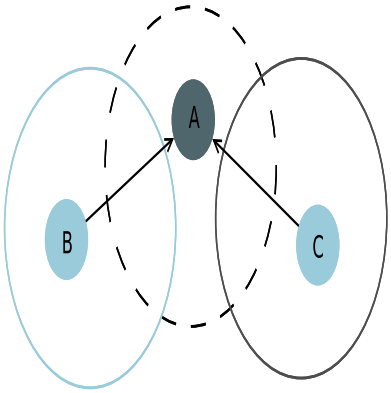

Figure 4.20: Hidden nodes

Table 4.12: Nodes' neighbors table.

Nodes	**Neighbors**
A	I, B, C
B	D, F, A
C	G, H, A
D	B, F
F	D, B, G
G	C, H, F
H	C, G
I	A

collision-free link (see Algorithm 19).

Algorithm 19 Checking the collision shared link state by a sender$_i$

VARIABLES
% link$_i$-state is a variable that indicates link$_i$ state
% NBC$_i$ is the number of communications scheduled in shared link$_i$
% NBR$_i$ is the number neighbors of each receiver R_{ik} scheduled in each shared link$_i$
% Neighbor$_{ik}$[NBR] is a table that stores the list of the nodes that belong to the influential range of a receiver R_{ik}.
% k the identification of a receiver scheduled in link$_i$
BEGIN
for < k := 1 to NBC$_i$ - 1 > **do**
  for < u := 1 to NBR$_i$ > **do**
    if (R_{ik} is neighbor$_{ik}$[u]) **then**
      if (R_{ij} belongs to the influential range of neighbor$_i k$[u]) **then**
        link$_i$-state = 'collision link'
      end if
    end if
  end for
end for
END

Complexity = ((NBC$_i$ - 1) × NBR$_i$× 3)

— NBC$_i$ - 1 is for the first loop

— NBR$_i$ is for the second loop

— 3: 2 for the if instructions and 1 for the assignment of the variable link$_i$

Note that complexity of diagnosing a shared link is $\mathbf{O}(NBC \times NBR)$.

Because the enhanced beacons in TSCH is repeated cyclically [34], and the schedule remains the same after the network formation, each sender executes Algorithm 19 just one time (during the first slotframe). Nodes (during other slotframes) only check the variable link$_i$-state of each link$_i$ to be aware of its state. In this case, the complexity of diagnosing a shared link become linear $\mathbf{O(1)}$.

4.3.2.2 Link quality Estimation and blacklisting techniques

When a collision happens on a collision shared link, the sender S_i deduces that this failure is due to an internal interference. In such a case, S$_i$ do not take this packet transmission failure into account in estimating the channel quality (see Algorithm 20).

Algorithm 20 Channel Quality Estimation

```
BEGIN
% NSi is the number of successful transmitted packets
% NFi is the number of failed transmitted packets in a collision-free
shared link or in a dedicated link
% NTi is the total number of transmitted packets in linki
if (Si sends packeti to Ri on linki) and (packeti is not acknowledged) then
  if (linki is a dedicated link) OR (linki is a collision-free shared link) then
    NF i ++ ;
    NT i ++;
  else
    NTi ++;
  end if
else
  if (packeti is acknowledged) OR (packeti is successfully received) then
    NS i ++;
    NTi ++;
  end if
end if
END
```

By counting the number of successfully transmitted packets, and the number of the failed transmitted packets due to external interference, every node should maintain

the Packet Reception Ration (PRR) of each link. The PRR helps the sender to know whether the channel should be blacklisted or whitelisted (See Algorithm 21). α is the acceptable link quality. We examined the α values to determine which value behaves best for our strategy (R-TSCH). The PRR is defined using equation 2.

$$PRR_i = \frac{NS_i}{(NS_i + NF_i)} \quad (2)$$

Where : $NS_i + NF_i \leq NT_i, i \in [0, 15]$

Algorithm 21 Blacklisting and whitelisting channels

```
BEGIN
if (PRR_i ≥ α) then
  if Link_i is already blacklisted then
    Whitelist Link_i;
  end if
else
  if Link_i is already whitelisted then
    Blacklist Link_i;
  end if
end if
END
```

4.4 Sixth Contribution: Enhanced Time Slotted Channel Hopping

We propose in this part a new dynamic blacklisting technique dedicated to TSCH mode to improve the reliability of communications. This part illustrates the problem statement, explains the link quality estimation process, describes the proposed Enhanced Time Slotted Channel Hopping (E-TSCH)and shows the simulation scenario. The obtained results are also analyzed.

4.4.1 Enhanced Time Slotted Channel Hopping

Each node in E-TSCH uses a dynamic mechanism, based on the channel conditions it estimates, to optimize the set of channels over which it switches. With this mechanism, a node avoids choosing channels that are degraded by interference such as from Wi-Fi devices. This dynamic channel selection mechanism spreads the traffic to the available channels. It aims to enhance communication robustness, to achieve sensor node energy efficiency, and packet delivery efficiency. To achieve such a goal, we

have proposed a new Intelligent Link Quality Estimation (I-LQE) process. The channel quality is estimated by taking into consideration the past and the current state of the channel, and detecting the channel condition by both the receiver and the sender scheduled on each channel. We also propose a new dynamic blacklisting algorithm to filter the best channels.

Assumptions:

we assume that:

- The network is already structured.
- All nodes are stationary (non mobile).
- The network has approximately a well-made schedule that achieves fairness between all nodes.
- The network supports the link asymmetry.

4.4.1.1 Intelligent Link Quality Estimation process I-LQE

According to the LQE process cited above, the I-LQE is proposed depending on the following characteristics:

— **Link monitoring:** for more accuracy, energy efficiency and less overhead purposes, our LQE employs link monitoring based on data traffic.

— **Link measurements:** for reactivity and dynamicity purposes, the link measurement of our proposal is based on sender-side (packet retransmission count) and receiver-side (acknowledgment count).

— **Metric evaluation:** the channel quality is estimated using the exponential smoothing technique (SES). The SES technique employs posterior statistical arguments to carefully evaluate the channel's quality to tightly exploit the channel's resources. The resources are also exploited by estimating the necessary duration that each channel should be blacklisted (temporal/permanent interference). The posterior statistical arguments are based on the probability that the current channel seems to be free of external interference.

The I-LQE algorithm preserves the energy of sensors and decreases the network traffic while estimating the channels conditions. It takes advantage of the link natures in TSCH (i.e. shared and dedicated links) to carefully estimate the channels quality. In

the case of failed transmission or reception of a packet, the I-LQE can simply deduce the cause of the problem being a simple internal collision accident or an external interference via the nature of the link on which the packet has failed to be transmitted.

The I-LQE classifies the shared links into two categories: collision-free shared links and collision shared links. The collision-free shared link is a link on which its participating nodes do not collide with each other. The collision shared link is a link on which its participating nodes belong to the scope of each other (e.g., hidden nodes).

To maintain this procedure, each sender S_i scheduled on each link holds a list of its colliding nodes and its receiver's colliding nodes. If a collision happens on a link l_i owing to the hidden node problem, the I-LQE process does not take such a transmission failure into account while estimating the quality of the link l_i.

To generate quality estimation for each channel c we use the following equation:

$$QE[c]_i \begin{cases} ChQ[c]_0 \quad , \quad i = 0. \\ \alpha QE[c]_{i-1} + (1-\alpha) ChQ[c]_i \quad , \quad i \in [1, \infty]. \end{cases} \tag{4.1}$$

$$ChQ[c]_i = \frac{ChGM}{(ChGM + ChBM)}.$$

$$c \in [11, 26] \quad , \quad \alpha \in [0, 1].$$

Where: c is the current channel, ChGM is the channels goodness metric and ChBM is the channels badness metric. $ChQ[c]_i$ and $QE[c]_i$ denote respectively the calculated percentage of ChGM and the obtained quality estimation of a channel c in the i^{th} instance. The initial estimation $QE[c]_0$ is assigned $ChQ[c]_0$, the first channel quality reading. The subsequent estimations at i are calculated using previous records $QE[c]_{i-1}$ and most recent channel quality $ChQ[c]_i$, subject to coefficient α.

The aim of the use of the $QE[c]_{i-1}$ variable is to tightly estimate the channels quality. We took the past quality estimation of the channel into consideration in order to know whether the current interference are temporal or permanent (e.g., mobile obstacles). So, if the interference is temporal, the channel can enter the blacklist just for a small period. Thus, we exploit efficiently the available resources.

If a node sends a data packet and receives an acknowledgment packet for it, the node increases the goodness metric (ChGM) of this channel by 1 and decrements the

value of its Repetitive Transmission Failure (RTF) by 1. If the node sends a data packet but does not receive an ACK for it on a dedicated link or on a collision-free shared link, the node increases the badness metric (ChBM) and the RTF of this channel by 1 (see algorithm 22); otherwise, the node assumes a packet collision may have occurred. Either way, the node increases the number of retries of the packet (NB) and starts the TSCH retransmission. The IEEE 802.15.4 e standard [34]limited the number of packet retransmissions, so that NB should not exceed *MacMaxFrameRetries*.

RTF ($0 \leq RTF \leq MaxRTF$) is a variable that indicates the number of failed packet transmission on channel c. RTF is incremented only by a sender $node_i$ when a $packet_k$ failed to be transmitted, and it is decremented only by a receiver $node_i$ when a $packet_k$ is correctly received on channel c. This variable is employed for channels blacklisting decisions purposes.

Algorithm 22 Sender node in E-TSCH

BEGIN
if (($node_i$ sends a $packet_k$ to $node_j$) and ($node_i$ receives an Ack_k from $node_j$)) **then**
  $ChGM \leftarrow ChGM + 1$
  $RTF[C] \leftarrow RTF[C] - 1$
else
  if (($packet_k$ is not acknowledged) and (($link_{ij}$ is dedicated) or ($link_{ij}$ is collision free shared link))) **then**
    $ChBM \leftarrow ChBM + 1$
    $RTF[C] \leftarrow RTF[C] + 1$
  end if
end if
End

When a node receives a packet, first it performs a validation test to determine whether the packet is acceptable or not. If the packet is approved, the node increases the goodness metric of this channel by 1, decrements the value of RTF by 1,and sends an ACK packet to the transmitter; otherwise, if it does not receive a valid packet on a dedicated link or on a collision-free shared link, the node increases the badness metric of this channel by 1 (see algorithm 23) .

By updating the channel badness and goodness metrics in this way, we ensure that only if a channel is congested with external interference, the badness metric of this channel increases; otherwise, the channel's goodness metric increases.

Algorithm 23 Receiver node in E-TSCH

BEGIN
if ($node_i$ receives a correct $packet_k$ from $node_j$) **then**
  Send Ack_k to $node_j$
  $ChGM \leftarrow ChGM + 1$
  $RTF[C] \leftarrow RTF[C] - 1$
else
  if (($link_{ij}$ is dedicated) or ($link_{ij}$ is collision free shared link)) **then**
    $ChBM \leftarrow ChBM + 1$
  end if
end if
End

4.4.1.2 Blacklisting Channels

Before sending a packet, the node performs a channel quality test to determine whether the channel should be blacklisted or not. A channel should be blacklisted if its estimated quality drops under the acceptable quality, or if it reaches the Max value of repetitive transmissions failure (MaxRTF) even though the estimated quality of this channel is above the acceptable quality. A channel is removed from the blacklist when its estimated quality rises above the acceptable quality and if its RTF is equal to 0.

The quality of a blacklisted channel can be enhanced, thus removed from the blacklist. Since the interference can be temporary, nodes can estimate different quality of the same channel. If channel C is blacklisted by $node_i$ but not by $node_j$, then $node_j$ can send and receive on this channel, while $node_i$ can only receive packets. When $node_i$ receives successful/valid packet from other nodes, $node_i$ increases the ChGM parameter of this channel, and hence the channel quality improves and eventually will be removed from the blacklist.

If all the channels are blacklisted, the node will sort the channels according to their estimated qualities (from highest to lowest), the top 4 channels are removed from the blacklist. Once a channel has been removed from the blacklist, a node uses this channel normally.

4.4.1.3 Channel Testing Algorithm

Before transmitting a packet, the node performs a channel quality test. Depending on the estimated quality and the RTF of the current channel, the node determines if this channel should be added/removed to/from the blacklist.

Algorithm 24 E-TSCH Channel Testing Algorithm

Variables:
C : the current channel.
$QE[\mathrm{C}]_i$: the current estimated quality for the channel C.
$RTF[\mathrm{C}]$: repeated transmission failure for a channel C.
MaxRTF : the maximum value of repeated transmission failure.
Acceptable : the acceptable quality threshold.
BEGIN
if (C is Blacklisted) **then**
  if (($QE[\mathrm{C}]_i \geq$ Acceptable) and ($RTF[\mathrm{C}] == 0$)) **then**
    remove C from the Blacklist.
  end if
else
  if ($QE[\mathrm{C}]_i \geq$ Acceptable) **then**
    if $RTF[\mathrm{C}] == MaxRTF$ **then**
      put C into the Blacklist.
    end if
  else
    put C into the Blacklist.
  end if
end if
End

Figure 4.21 shows two possible scenarios for a node in E-TSCH, after channel sampling and continuously updating channels metrics; the node makes the decision of blacklisting a channel or removing it from the blacklist. In this example, we define the Acceptable Quality and the Max Repetitive Transmission Failure as follow:

- Acceptable Quality = 0.7
- MaxRTF = 3

- **Channel 1:** : a node blacklists channel 1, when the value of RTF reaches *MaxRTF* (3). The channel is blacklisted even if its quality is above the acceptable quality. Successive transmission failures (RTF) may indicate that the channels risks to be altered by momentary external interference.
- **Channel 2:** when the node receives successive correct packets, it decrements the value of RTF (the channel quality improves over time). To ensure that the channel is reliable, it should not be removed from the blacklist unless its estimated quality rises above the acceptable quality and its RTF is equal to 0.

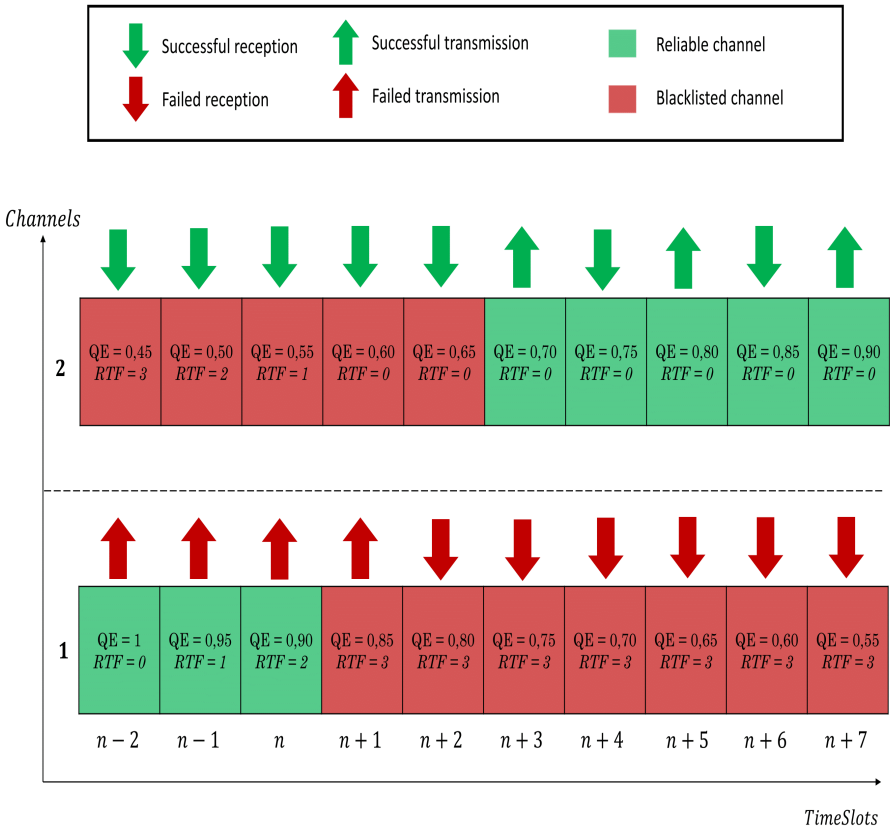

Figure 4.21: Example of blacklisting / unblacklisting scenarios in E-TSCH

4.4.1.4 Simulation Scenario

To measure the potential performance improvements that can be achieved when using E-TSCH instead of TSCH, we performed a set of simulation experiments using the NS3 simulation tool [110]. We first implemented the NS3 modules needed to build an IEEE 802.15.4 network and then the NS3 modules for a TSCH network. In our experiments, we considered a sensor network with greed topology, where the sink node acts as the PAN coordinator and 10 sensor nodes are placed around the PAN coordinator, simulation parameters are shown in Table 4.13. We considered a periodic reporting application where data acquired by sensors have to be reported periodically to the PAN coordinator.

Previous results [111] [112] [113] [105] [114] clearly show that the presence of Wi-Fi signals greatly degrades the IEEE 802.15.4 network performance, in terms of throughput, delay, and energy consumption. With that in mind, an IEEE 802.11-2007 (Wi-Fi)-compliant device is placed near the network to act as an interference source, it is configured to generate Wi-Fi waves every 2 ms, making it almost impossible for nodes

Parameter	**Value**
Duration	3600 s
Timeslot duration	10 ms
Initial energy per node	2 J
Initial supply voltage	3 V
Radio Tx current	0.007 A
Radio Rx current	0.0005 A
Radio Idle current	0.00000052 A
Max Frame Retries	5 time slots

Table 4.13: Simulation parameters

to communicate on the affected channels.

A set of experiments were carried out with different configurations for 1 hour duration (3600 seconds) to determine which configuration works best for E-TSCH. Based on these experiments we defined the MaxRTF by 3 and the Acceptable Quality by 0.7.

Evaluation Metrics

To evaluate the performance of the two protocols (TSCH and E-TSCH), we derived the following evaluation metrics:

- **Packet delivery ratio (PDR) (%):** defined as the ratio between the number of data packets correctly received and the total number of data packets generated by the network. It measures the network reliability in the data collection process.
- **Energy per packet (Joules):** defined as the total energy consumed by each sensor node divided by the number of data packets correctly delivered to the PAN coordinator. It measures the energy efficiency of the system.
- **Dropped packets:** defined as the total number of dropped packets by the network (i.e., a packet is dropped when it reaches the *MaxFramesRetries*. *MaxFramesRetries* is the maximum allowed number of retransmissions of a packet [34]).
- **Retransmissions:** defined as the total number of occurred retransmissions of data packets in the network.

Results and Analysis

First, we discuss the experiments results to determine the best configurations of E-TSCH. Then we perform a comparison between the two protocols.

4.4.1.4.1 Performance Comparison As expected, the presence of a Wi-Fi device in the simulated scenario strongly influences the sensor network's behavior.

Figure 4.22 represents the number of the transmitted packets of both protocols with respect to time. Notice that E-TSCH transmitted 36480 packets while TSCH transmitted just 22605 packets. Such a behavior is a result of the blind channel-hopping nature of TSCH, as it keeps retransmitting packets on undesirable channels and eventually drops them (Figures 4.23 and 4.24). The gap between the two protocols grows over time, as the difference is 13875 packets at time 3600 s.

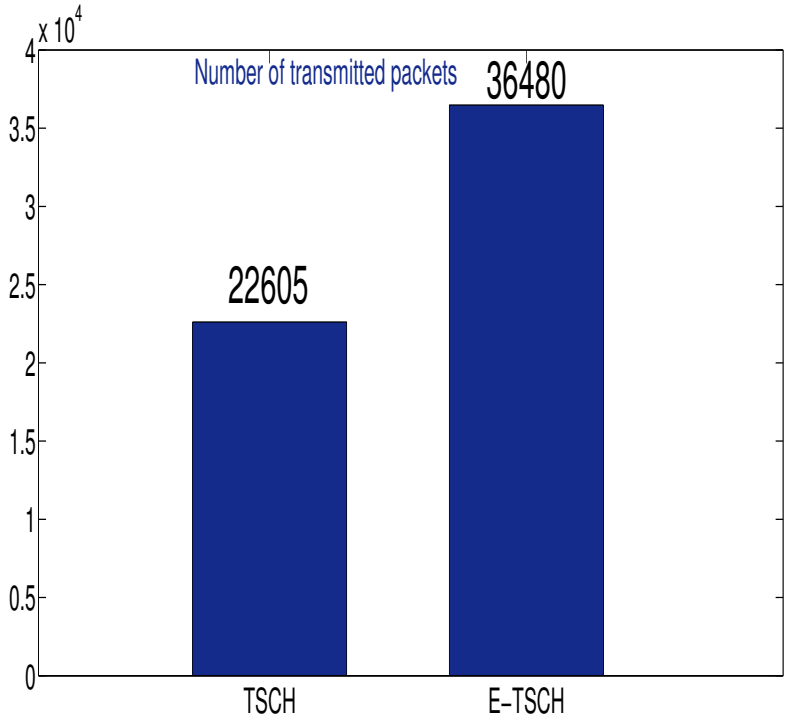

Figure 4.22: Number of transmitted packets of both TSCH and E-TSCH.

To further demonstrate the positive effect of E-TSCH, we have traced the momentary PDR values upon receipts of each packet during the communications and compiled such fluctuations in Figure 4.25 .

Figure 4.25 shows that E-TSCH performs better after approximately 50 s from the start of the experiment, as the delivery ratio of E-TSCH is predominantly above 90% after 500 s compared with the descending values of TSCH. Moreover, E-TSCH shows a steady trend of growth. This is because blacklist decisions become increasingly accurate, as continuing communications allows for more channel-quality readings to be collected.

Figure 4.26 shows the performance of the two protocols in terms of energy con-

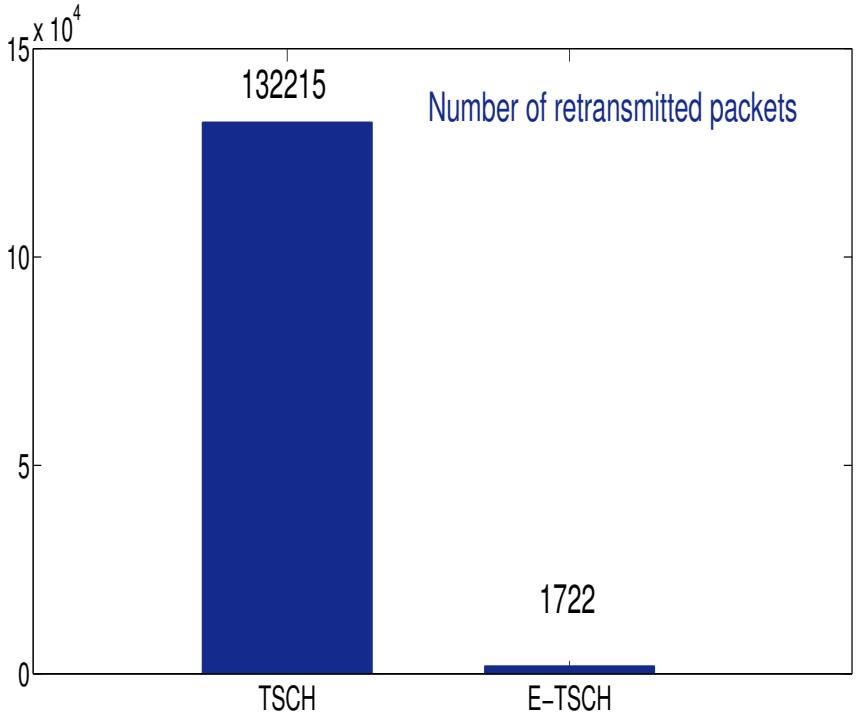

Figure 4.23: Number of retransmitted packets.

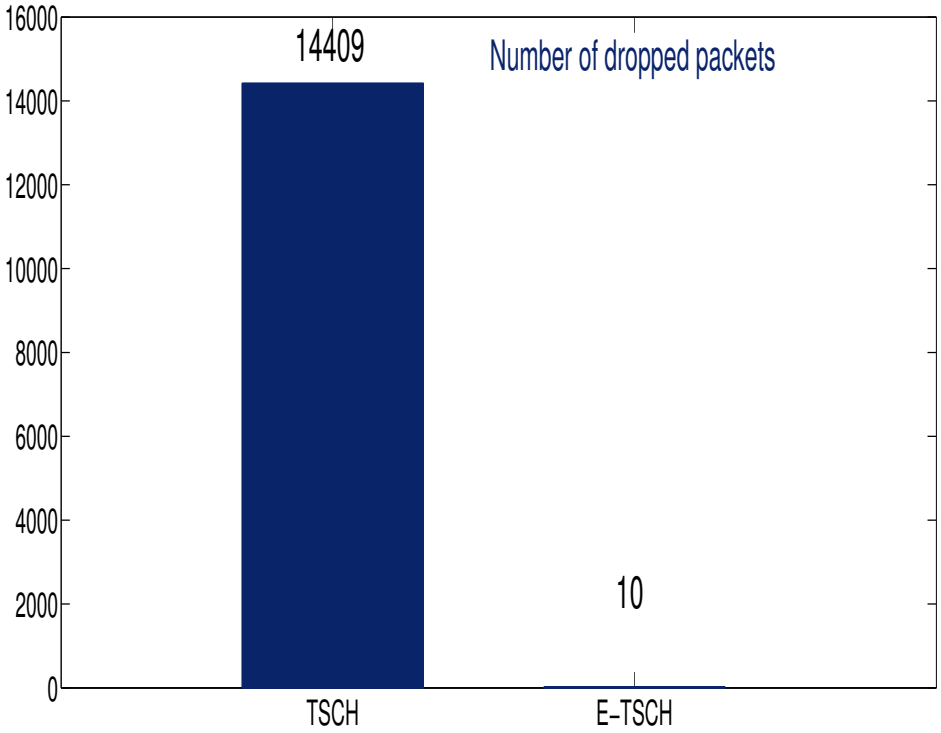

Figure 4.24: Number of dropped packets.

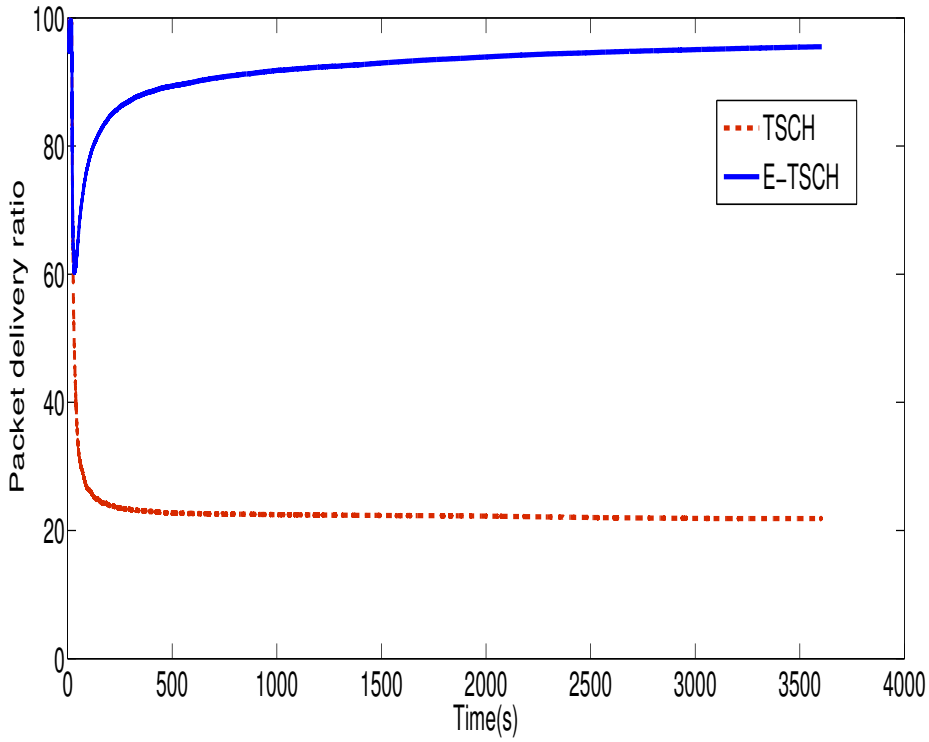

Figure 4.25: Packet delivery ratio vs. time.

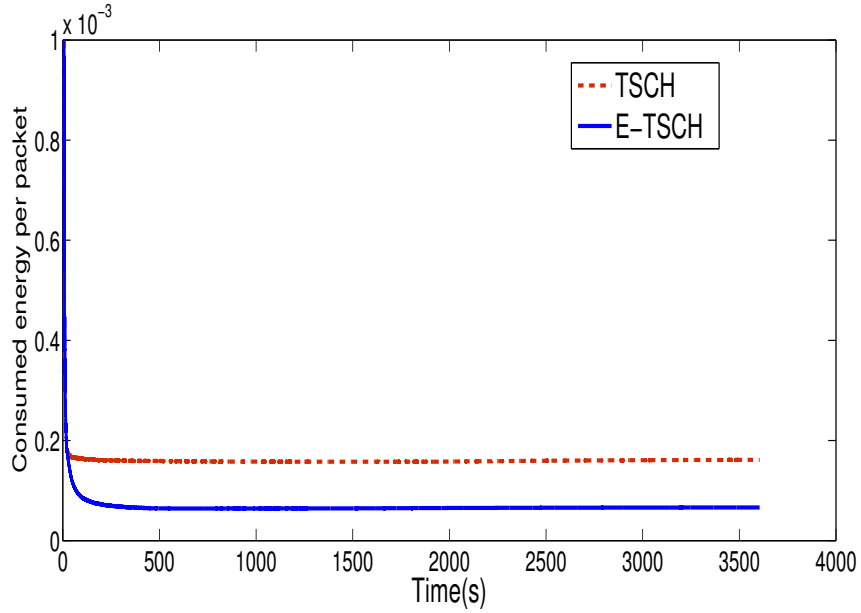

Figure 4.26: packet energy consumption vs. time.

sumption. Specifically, the per-packet energy consumption is provided to show the effectiveness of the two protocols when managing the energy source. As expected, E-TSCH outperforms TSCH due to the high number of retransmissions. Note that TSCH

spends more energy over time, unlike E-TSCH, for which, after approximately 500 s, the consumed energy becomes somewhat constant.

4.5 Conclusion

The use of WSNs in industrial environments presents challenges, because of the presence of typical interference sources like microwave ovens and Wi-Fi devices etc... These can cause high destructive interference. Also, the industrial environment usually contains metallic and mobile objects, such as robots, cars and people, which usually cause multi-path fading effects. In addition to external interference, internal interference also can drastically deteriorate the communication performance. Combined with the fact that wireless links are inherently unreliable, and suffer from interference in the spectral band used for communication, makes it difficult to improve the quality of service. Collisions may seriously disturb the critical applications deployed in low-power lossy networks. They decrease the network lifetime since it consumes a lot of energy when retransmitting packets. It also increases the packet latency owing to the appliance of the TSCH CSMA-CA backoff and PCA backoff retransmission algorithms executed for the packet retransmission. To achieve a highest communication reliability, we addressed in chapter four the following contributions.

— The first contribution presents two intelligent algorithms that provide collision avoidance in TSCH mode. The first backoff algorithm is called Time Slotted Channel Hopping with Correct Collision Avoidance (TSCH-CCA). The Second one is called Enhanced Priority Channel Access Backoff Algorithm (E-PCA). The TSCH-CCA is applied for ordinary packets, while the E-PCA is applied for critical event packets. The proposed solutions are compared respectively with the TSCH CSMA-CA backoff algorithm and with the PCA backoff algorithm. The results showed significant enhancements in terms of latency, network congestion, network lifetime, critical event packets lifetime, and collision avoidance.

— We proposed in the second contribution a new link quality estimation process dedicated to TSCH. The technique whitelists the links with high quality and blacklists the worst ones. We implemented our approach and TSCH using NS3 and run a number of experiments. The results show significant improvements in

terms of energy consumption, packet delivery ratio, throughput and total number of packets retransmission.

— In third contribution, we proposed a link quality estimation (I-LQE) process. We also proposed an efficient blacklisting technique that avoids using undesirable channels in the Time Slotted Channel Hopping (TSCH) mode. The proposed solution has been implemented using the Network Simulator 3. It has been tested with different configurations. Compared to the blind IEEE 802.15.4e TSCH mode, our proposed methods show significant improvements in terms of reliability, energy efficiency and throughput.

Conclusion and Future Work

The emerging idea of the Internet of Things (IoT) is rapidly finding its path throughout our modern life, aiming to improve the quality of life by connecting many smart devices, technologies, and applications. Wireless sensor networks and Cloud computing are the main technologies that enabled the growth of IoT. Despite the valuable services that these elements offer, they still have some challenges that should be treated, otherwise the quality of services of IoT applications will negatively deteriorate. Preserving energy consumption, increasing the network longevity, decreasing latency time and fault tolerance are the most concerns that should be taken into consideration when building IoT applications. From the literature review discussed in chapter one and chapter two, we conclude that MAC-sublayer protocols can achieve a high energy gain of sensor, thus ensure the longevity of the network when they are efficiently designed to overcome internal interface, external interference and multi-path fading. It can also achieve minimum latency time when data are successfully transmitted for the first attempt, because data retransmission also increases the end to end packet delivery latency. Despite the efficiency that TSCH offers to WSNs, it still has some drawbacks that should be treated to reach a higher efficiency and a good quality of services to IoT applications. It has two main problems: interference (internal and external) and multi-path fading. The external interference comes from the fact that it uses all channels indiscriminately while channels are subjected to different level of interference. The internal interference occurs when two colliding nodes transmit their data at the same time. To overcome internal interference in the presence of hidden nodes, this thesis proposes two intelligent algorithms [108] (Time Slotted Channel Hopping with Correct Collision Avoidance backoff algorithm(TSCH-CCA) and Enhanced Priority Channel Access Backoff Algorithm(E-PCA)) applied respectively to both normal packets and

critical events packets. Our proposed solution showed significant improvements in terms of latency, network congestion, network lifetime, critical event packets lifetime, and collision avoidance. To face the external interference and multi-path problems, we proposed new strategies dedicated to TSCH that makes sensors intelligent and enables them to hop channels in a wise manner. The strategies present a new link quality estimator used to identify and exclude undesirable channels that are congested with Wi-Fi interference. We evaluated the performance of our proposals that overcome the external interference and multi-path fading, we implemented our techniques in Network Simulator 3 (NS3) and compared them to TSCH. The results showed significant improvements in terms of number of retransmitted packets, packet delivery ratio, and energy consumption.

In addition to MAC-sublayer, Cloud computing also plays a vital role in the data input/output latency. Cloud computing allows users to create their IaaS to store and control the storage. The unstructured stored data can significantly increase the latency of clients requests. In this thesis, we performed an efficient input and output requests. We proposed 3 contributions to build an IaaS based on load balancing technique to distribute the load evenly among the servers. We improved Big Data structuring, addressed the bad storage, input/output response time challenges and achieved a fault tolerant and resilient system by providing critical data availability at any moment with minimum resources exploitation. Our contributions are based on parallelism, and collaboration of multi-agents system to achieve minimum latency. We implemented them using JADE Platform(Java). The results showed significant improvements in terms of latency of critical data requests and critical data availability.

As a future work we plan to:

— publish a paper entitled “An intelligent Clustering and scheduling techniques for TSCH” in a journal. The proposed technique enhances the reliability of data transmission in shared links, it also increases the throughput, minimizes the latency and preserves the energy of sensors.

— implement the second proposal and publish it in a journal paper. This work is about a new blacklisting algorithm to detect or predict the source of a transmission failure (a multi-path fading or an external interference like Wifi).

— design an approach to secure the IaaSs to protect the critical data stored on

Cloud servers.

4.5.1 Contributions

— First contribution seeks to explore the challenges of IoT. It enables us to understand more the field, define its challenges and work on the problematic of our thesis as defined in chapter one [5].

— Second contribution achieves load balancing of Big data on Cloud computing servers [7].

— Third contribution realizes an intelligent storage and critical data redundancy with minimum resources exploitation. Beside to this benefits, it provides system monitoring and data availability within minimum latency [101].

— Fourth contribution validates the third contribution under JADE platform and solves the problem of single point failure that the third contribution suffers from [115].

— Fifth contribution addresses WSNs challenges. It ameliorates the TSCH mode proposed by IEEE 802.15.4e by eliminating internal collisions in the presence of hidden nodes. We proposed two intelligent algorithms (Time Slotted Channel Hopping with Correct Collision Avoidance backoff algorithm(TSCH-CCA) and Enhanced Priority Channel Access Backoff Algorithm(E-PCA)) applied respectively to both normal packets and critical events packets [108].

— Sixth contribution provides a solution that solves external interference and multi-path fading in WSNs deployed in harsh and incompatible wireless technologies co-existence environment [109].

— Seventh contribution also solves external interference and multi-path fading. It uses the past and the present state of channels to tightly estimate the links quality.

Bibliography

[1] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[2] Ivanovitch Silva, Rafael Leandro, Daniel Macedo, and Luiz Affonso Guedes. A dependability evaluation tool for the internet of things. *Computers & Electrical Engineering*, 39(7):2005–2018, 2013.

[3] Shanzhi Chen, Hui Xu, Dake Liu, Bo Hu, and Hucheng Wang. A vision of iot: Applications, challenges, and opportunities with china perspective. *IEEE Internet of Things journal*, 1(4):349–359, 2014.

[4] In Lee and Kyoochun Lee. The internet of things (iot): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4):431–440, 2015.

[5] Sarra Hammoudi, Zibouda Aliouat, and Saad Harous. Challenges and research directions for internet of things. *Telecommunication Systems*, 67(2):367–385, 2018.

[6] Vehbi C Gungor, Gerhard P Hancke, et al. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *IEEE Trans. Industrial Electronics*, 56(10):4258–4265, 2009.

[7] Sarra Hammoudi, Abdelhafid Benaouda, Saad Harous, and Zibouda Aliouat. Load balancing in the cloud using specialization. In *Ubiquitous Computing, Electronics & Mobile Communication Conference*, pages 1–7. IEEE, 2016.

[8] Zaid Ali Shhedi. A real-time monitoring system for hospital acquired infection prevention (sistem de monitorizare în timp real pentru prevenirea infecţiilor intraspitaliceti).

[9] Jing Wang and Xiaola Lin. User dispersed authentication scheme for wireless sensor networks. In *Digital Home (ICDH), 2012 Fourth International Conference on*, pages 411–414. IEEE, 2012.

[10] R Nithya and N Mahendran. A survey: Duty cycle based routing and scheduling in wireless sensor networks. In *Electronics and Communication Systems (ICECS), 2015 2nd International Conference on*, pages 813–817. IEEE, 2015.

[11] R Maheswar, P Jayarajan, and F Nathirulla Sheriff. A survey on duty cycling schemes for wireless sensor networks. *International Journal of Computer Networks and Wireless Communications*, 3(1):37–40, 2013.

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

[13] Peng Du and George Roussos. Adaptive time slotted channel hopping for wireless sensor networks. In *Computer Science and Electronic Engineering Conference (CEEC), 2012 4th*, pages 29–34. IEEE, 2012.

[14] Ruan D Gomes, Gláucio B Rocha, Abel C Lima Filho, Iguatemi E Fonseca, and Marcelo S Alencar. Distributed approach for channel quality estimation using dedicated nodes in industrial wsn. In *Personal, Indoor, and Mobile Radio Communication (PIMRC), 2014 IEEE 25th Annual International Symposium on*, pages 1943–1948. IEEE, 2014.

[15] Per Ängskog, Carl Karlsson, Javier Ferrer Coll, José Chilo, and Peter Stenumgaard. Sources of disturbances on wireless communication in industrial and factory environments. In *Electromagnetic Compatibility (APEMC), 2010 Asia-Pacific Symposium on*, pages 281–284. IEEE, 2010.

[16] Ruan Delgado Gomes, Marco Aurelio Spohn, Abel Cavalcante Lima, Eudisley Gomes dos Anjos, and Francisco Antonio Belo. Correlation between spectral occupancy and packet error rate in ieee 802.15. 4-based industrial wireless sensor networks. *IEEE Latin America Transactions*, 10(1):1312–1318, 2012.

[17] Wenqi Guo, William M Healy, and MengChu Zhou. Impacts of 2.4-ghz ism band interference on ieee 802.15. 4 wireless sensor network reliability in buildings. *IEEE Transactions on Instrumentation and Measurement*, 61(9):2533–2544, 2012.

[18] Abel C Lima-Filho, Ruan D Gomes, Marceu O Adissi, Tassio Alessandro Borges Da Silva, Francisco A Belo, and Marco A Spohn. Embedded system integrated into a wireless sensor network for online dynamic torque and efficiency monitoring in induction motors. *IEEE/ASME transactions on mechatronics*, 17(3):404–414, 2012.

[19] Micrium. Part 1: IoT Devices and Local Networks — Micrium. `https://www.micrium.com/iot/devices/`.

[20] Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things: a survey. *Information Systems Frontiers*, 17(2):243–259, 2015.

[21] Internet of Things and Future Internet Enterprise Systems. `http://cordis.europa.eu/fp7/ict/enet/rfid-iot_en.html`. Accessed: 2010-09-30.

[22] J. Philip and Jr Koopman. Embedded system design issues (the rest of the story). In *Proceedings of the International Conference on Computer Design (ICCD 96) in Computers and Processors*, page 310. IEEE, 1996.

[23] D INFSO. Networked enterprise & rfid infso g. 2 micro & nanosystems, in co-operation with the working group rfid of the etp eposs, internet of things in 2020, roadmap for the future [r]. *Information Society and Media, Tech. Rep*, 2008.

[24] ITU. Itu: Committed to connecting the world. `http://www.itu.int/en/ITUT/gsi/iot/Pages/default.aspx`. Accessed: 2016.

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

[26] Dave Evans. The internet of things: How the next evolution of the internet is changing everything. *CISCO white paper*, 1(2011):1–11, 2011.

[27] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. Future internet: the internet of things architecture, possible applications and key challenges. In *Frontiers of Information Technology (FIT), 2012 10th International Conference on*, pages 257–260. IEEE, 2012.

[28] Luis Felipe Herrera-Quintero, Francisco Maciá-Pérez, Diego Marcos-Jorquera, and Virgilio Gilart-Iglesias. Wireless sensor networks and service-oriented ar-

chitecture, as suitable approaches to be applied into its. In *Proceedings of the 6th Euro American Conference on Telematics and Information Systems*, pages 301–308. ACM, 2012.

[29] Karl Aberer, Manfred Hauswirth, and Ali Salehi. The global sensor networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical report, 2006.

[30] datatracker, Constrained RESTful Environments (core). `https://datatracker.ietf.org/wg/core/charter/`.

[31] Micrium. IoT Standards and Protocols. `http://www.postscapes.com/internet-of-things-protocols/`.

[32] Walter Colitti, Kris Steenhaut, Niccolò De Caro, Bogdan Buta, and Virgil Dobrota. Evaluation of constrained application protocol for wireless sensor networks. In *Local & Metropolitan Area Networks (LANMAN), 2011 18th IEEE Workshop on*, pages 1–6. IEEE, 2011.

[33] Jonathan W Hui and David E Culler. Extending ip to low-power, wireless personal area networks. *IEEE Internet Computing*, (4):37–45, 2008.

[34] Ieee standard for low-rate wireless networks. *IEEE Std 802.15.4 TM-2015*, 2015.

[35] Hung-Cuong Le, Violeta Felea, et al. Obmac: an overhearing based mac protocol for wireless sensor networks. In *sensorcomm*, pages 547–553. IEEE, 2007.

[36] Domenico De Guglielmo, Simone Brienza, and Giuseppe Anastasi. Ieee 802.15. 4e: A survey. *Computer Communications*, 88:1–24, 2016.

[37] Timothy R Petty, Nawajish Noman, Deng Ding, and John B Gongwer. Flood forecasting gis water-flow visualization enhancement (wave): A case study. *Journal of Geographic Information System*, 8(06):692, 2016.

[38] pubnub. `https://www.pubnub.com/`.

[39] Part 1: IoT Devices and Local Networks — Micrium. `https://www.adafruit.com/product/2718`.

[40] Sensor Web Enablement (SWE). `http://www.opengeospatial.org/ogc/markets-technologies/swe`. Accessed: 16-03-2017.

[41] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future Generation Computer Systems*, 56:684–700, 2016.

[42] Barnatt C. Cloud Computing. `http://explainingcomputers.com/cloud.html`. Accessed: 2016-05-14.

[43] Zeeshan Amin, Harshpreet Singh, and Nisha Sethi. Review on fault tolerance techniques in cloud computing. *International Journal of Computer Applications*, 116(18), 2015.

[44] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[45] John A Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1):3–9, 2014.

[46] Arun Kanuparthi, Ramesh Karri, and Sateesh Addepalli. Hardware and embedded security in the context of internet of things. In *Proceedings of the 2013 ACM workshop on Security, privacy & dependability for cyber vehicles*, pages 61–64. ACM, 2013.

[47] Yunchuan Sun, Houbing Song, Antonio J Jara, and Rongfang Bie. Internet of things and big data analytics for smart and connected communities. *IEEE Access*, 4:766–773, 2016.

[48] Hsieh-Chung Chen, Harnek Gulati, HT Kung, and Surat Teerapittayanon. Compressive wireless pulse sensing. In *Collaboration Technologies and Systems (CTS), 2015 International Conference on*, pages 5–11. IEEE, 2015.

[49] Barnatt C. Big Data. `http://www.explainingcomputers.com/big_data.html`. Accessed: 2016-05-07.

[50] Enzo Baccarelli, Nicola Cordeschi, Alessandro Mei, Massimo Panella, Mohammad Shojafar, and Julinda Stefa. Energy-efficient dynamic traffic offloading and reconfiguration of networked data centers for big data stream mobile computing: review, challenges, and a case study. *IEEE Network*, 30(2):54–61, 2016.

[51] Rajwinder Kaur and Pawan Luthra. Load balancing in cloud computing. In *Proceedings of International Conference on Recent Trends in Information, Telecommunication and Computing, ITC*. Citeseer, 2012.

[52] Sairam Vakkalanka. A classification of job scheduling algorithms for balancing load on web servers. *International Journal of Modern Engineering Research (IJMER)*, 2(5):3679–3683, 2012.

[53] Tony Bourke. *Server load balancing*. " O'Reilly Media, Inc.", 2001.

[54] S Sharma and J Godara. Load balancing in cloud computing. *International Journal of Computer Systems*, 3(4):322–326, 2016.

[55] Ishita Patel and Brona Shah. Survey on resource allocation technique in cloud. *Int. J. Sci. Res*, 5:232–235, 2016.

[56] Snehlata Mishra and Ritu Tondon. A shared approach of dynamic load balancing in cloud computing. *International Journal of Scientific Research in Science. Engineering and Technology (ijsrset. com)*, 2(02):632–638, 2016.

[57] Vijay Kumar and Jitender Kumar. A comparative study of load balancing techniques in cloud computing environment. *International Journal of Innovations & Advancement in Computer Science*, 5(4):29–32, 2016.

[58] Sandeep Sharma, Sarabjit Singh, and Meenakshi Sharma. Performance analysis of load balancing algorithms. *World Academy of Science, Engineering and Technology*, 38(3):269–272, 2008.

[59] Monika Kushwaha and S Gupta. Various schemes of load balancing in distributed systemsa review. *International Journal of Scientific Research Engineering & Technology (IJSRET)*, 4(7):741–748, 2015.

[60] Soumya Ray and Ajanta De Sarkar. Execution analysis of load balancing algorithms in cloud computing environment. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 2(5):1–13, 2012.

[61] Mohammad Reza Mesbahi, Mahnaz Hashemi, and Amir Masoud Rahmani. Performance evaluation and analysis of load balancing algorithms in cloud computing environments. In *Web Research (ICWR), 2016 Second International Conference on*, pages 145–151. IEEE, 2016.

[62] Hervé Guyennet, Bénédicte Herrmann, Laurent Philippe, and François Spies. A performance study of dynamic load balancing algorithms for multicomputers. In *Massively Parallel Computing Systems, 1994., Proceedings of the First International Conference on*, pages 538–540. IEEE, 1994.

[63] Poonam Devi and Trilok Gaba. Implementation of cloud computing by using short job scheduling. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(7):178–183, 2013.

[64] Valeria Cardellini, Michele Colajanni, and Philip S Yu. Dynamic load balancing on web-server systems. *IEEE Internet computing*, 3(3):28–39, 1999.

[65] Srinivas Sethi, Anupama Sahu, and Suvendu Kumar Jena. Efficient load balancing in cloud computing using fuzzy logic. *IOSR Journal of Engineering*, 2(7):65–71, 2012.

[66] Tushar Desai and Jignesh Prajapati. A survey of various load balancing techniques and challenges in cloud computing. *International Journal of Scientific & Technology Research*, 2(11):158–161, 2013.

[67] Ranjita Bhagwan, Stefan Savage, and Geoffrey M Voelker. Understanding availability. In *International Workshop on Peer-to-Peer Systems*, pages 256–267. Springer, 2003.

[68] Nourhene Maalel, Enrico Natalizio, Abdelmadjid Bouabdallah, Pierre Roux, and Mounir Kellil. Reliability for emergency applications in internet of things. In *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference On*, pages 361–366. IEEE, 2013.

[69] Dewei Yi and Huanjia Yang. Heer–a delay-aware and energy-efficient routing protocol for wireless sensor networks. *Computer Networks*, 104:155–173, 2016.

[70] Xing Shao, C Wang, and Yuan Rao. Network coding aware qos routing for wireless sensor network. *Journal of Communications*, 10(1):24–32, 2015.

[71] Deng-ao Li, Hailong Hao, Guolong Ji, and Jumin Zhao. An adaptive clustering algorithm based on improved particle swarm optimisation in wireless sensor networks. *International Journal of High Performance Computing and Networking*, 8(4):370–380, 2015.

[72] Technical.openmobilealliance.org. `http://technical.openmobilealliance.org/Technical/technicalinformation/release-program/current-releases/omalightweightm2mv1-0-2`. Accessed: 2016-05-07.

[73] Stefan Wallin and Claes Wikström. Automating network and service configuration using netconf and yang. In *LISA*, 2011.

[74] Plugtests events. `http://www.etsi.org/about/what-we-do/plugtests`. Accessed: 2016-08-07.

[75] Pursuing ROadmaps and BEnchmarks for the Internet of Things 2013. `http://www.probe-it.eu/`. Accessed: 2016-03-07.

[76] Kris Bubendorfer and John H Hine. *A compositional classification for load-balancing algorithms*. School of Mathematical and Computing Sciences, Victoria University of Wellington, 1999.

[77] Bellabas Yagoubi. Modèle d'équilibrage de charge pour les grilles de calcul. *Revue Africaine de la Recherche en Informatique et Mathématiques Appliquées*, 7:1–19, 2007.

[78] Lekha Nema, Avinash Sharma, and Saurabha Jain. Load balancing algorithms in cloud computing: An extensive survey. *International Journal of Engineering Science and Computing*, 6(6), 2016.

[79] Pradeep Naik, Surbhi Agrawal, and Srikanta Murthy. A survey on various task scheduling algorithms toward load balancing in public cloud. *American Journal of Applied Mathematics*, 3(1-2):14–17, 2015.

[80] Chunsheng Zhu, Victor CM Leung, Laurence T Yang, Lei Shu, Joel JPC Rodrigues, and Xiuhua Li. Trust assistance in sensor-cloud. In *Computer Communications Workshops (INFOCOM WKSHPS), 2015 IEEE Conference on*, pages 342–347, 2015.

[81] Reinhard Maier. Event-triggered communication on top of time-triggered architecture. In *Digital Avionics Systems Conference, 2002. Proceedings. The 21st*, volume 2, pages 13C5–13C5. IEEE, 2002.

[82] Zibouda Aliouat and Makhlouf Aliouat. Verification of cooperative transient fault diagnosis and recovery in critical embedded systems. *Int. Arab J. Inf. Technol.*, 9(4):373–381, 2012.

[83] Hermann Kopetz. Event-triggered versus time-triggered real-time systems. In *Operating Systems of the 90s and Beyond*, pages 86–101. Springer, 1991.

[84] Manuel Mazo and Paulo Tabuada. On event-triggered and self-triggered control over sensor/actuator networks. In *Decision and Control, 47th IEEE Conference on*, pages 435–440. IEEE, 2008.

[85] Michael Wooldridge. An introduction to multi agent systems, department of computer science, university of liverpool, uk, 2002.

[86] Walter Brenner, Rüdiger Zarnekow, and Hartmut Wittig. Agents as tools of the information society. In *Intelligent Software Agents*, pages 7–18. Springer, 1998.

[87] Nicholas N Karekwaivanane, Wilson Bakasa, and Kudakwashe Zvarevashe. Reliability in mac protocols for wireless sensor networks: A survey, 2014.

[88] Yang Yang and Weidong Yi. Rainbow: Reliable data collecting mac protocol for wireless sensor networks. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6. IEEE, 2010.

[89] Pedro Henrique Gomes, Thomas Watteyne, Pradipta Ghosh, and Bhaskar Krishnamachari. Competition: Reliability through timeslotted channel hopping and flooding-based routing. In *EWSN*, pages 297–298, 2016.

[90] Asimakis Tzamaloukas and JJ Garcia-Luna-Aceves. Channel-hopping multiple access. In *Communications, 2000. ICC 2000. 2000 IEEE International Conference on*, volume 1, pages 415–419. IEEE, 2000.

[91] HwaMin Lee, Se Dong Min, Min-Hyung Choi, and DaeWon Lee. Multi-agent system for fault tolerance in wireless sensor networks. *KSII Transactions on Internet & Information Systems*, 10(3), 2016.

[92] Nouha Baccour, Anis Koubâa, Luca Mottola, Marco Antonio Zúñiga, Habib Youssef, Carlo Alberto Boano, and Mário Alves. Radio link quality estimation in wireless sensor networks: A survey. *ACM Transactions on Sensor Networks (TOSN)*, 8(4):34, 2012.

[93] N Baccour, A Koubaa, M Jamaa, H Youssef, and M Zuniga. A comparative simulation study of link quality estimators in wsn. *MASCOTS'09. IEEE*, 2009.

[94] Kihoon Jeon and Sanghwa Chung. Adaptive channel quality estimation method for enhanced time slotted channel hopping on wireless sensor networks. In *Ubiq-*

uitous and Future Networks (ICUFN), 2017 Ninth International Conference on, pages 438–443. IEEE, 2017.

[95] Kris Pister and Lance Doherty. Tsmp: Time synchronized mesh protocol. *IASTED Distributed Sensor Networks*, 391:398, 2008.

[96] Domenico De Guglielmo, Simone Brienza, and Giuseppe Anastasi. Ieee 802.15. 4e: A survey. *Computer Communications*, 88:1–24, 2016.

[97] Junaid Ansari and Petri Mähönen. Channel selection in spectrum agile and cognitive mac protocols for wireless sensor networks. In *Proceedings of the 8th ACM international workshop on Mobility management and wireless access*, pages 83–90. ACM, 2010.

[98] Peng Du and George Roussos. Adaptive channel hopping for wireless sensor networks. In *Mobile and Wireless Networking (iCOST), 2011 International Conference on Selected Topics in*, pages 19–23. IEEE, 2011.

[99] Jangkyu Yun, Byeongjik Lee, Jilong Li, and Kijun Han. A channel switching scheme for avoiding interference of between ieee 802.15. 4 and other networks. In *Computer and Computational Sciences, 2008. IMSCCS'08. International Multi-symposiums on*, pages 136–139. IEEE, 2008.

[100] G Siva Shanmugam and N Ch SN Iyengar. Effort of load balancer to achieve green cloud computing: A review. *International Journal of Multimedia and Ubiquitous Engineering*, 11(3):317–332, 2016.

[101] Sarra Hammoudi, Zibouda Aliouat, and Saad Harous. A new infrastructure as a service for iot-cloud. In *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pages 786–792. IEEE, 2018.

[102] Pascal Chatonnay, Bénédicte Herrmann, and Laurent Philippe. Etude comparative d'algorithmes d" equilibrage de charge sur multicalculateurs. 1999.

[103] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12:493–506, 2004.

[104] N Azmi, LM Kamarudin, M Mahmuddin, A Zakaria, AYM Shakaff, S Khatun, K Kamarudin, and MN Morshed. Interference issues and mitigation method

in wsn 2.4 ghz ism band: A survey. In *Electronic Design (ICED), 2014 2nd International Conference on*, pages 403–408. IEEE, 2014.

[105] Kannan Srinivasan, Prabal Dutta, Arsalan Tavakoli, and Philip Levis. An empirical study of low-power wireless. *ACM Transactions on Sensor Networks (TOSN)*, 6(2):16, 2010.

[106] Axel Sikora and Voicu F Groza. Coexistence of standardized wireless systems in the 2.4 ghz-ism-band. *International Journal of Computing*, 4(1):5–11, 2014.

[107] Domenico De Guglielmo, Giuseppe Anastasi, and Alessio Seghetti. From ieee 802.15. 4 to ieee 802.15. 4e: A step towards the internet of things. In *Advances onto the Internet of Things*, pages 135–152. Springer, 2014.

[108] Sarra Hammoudi, Saad Harous, Zibouda Aliouat, and Lemia Louail. Time slotted channel hopping with collision avoidance. *International Journal of Ad Hoc and Ubiquitous Computing*, 29(1-2):85–102, 2018.

[109] Sarra Hammoudi, Saad Harous, and Zibouda Aliouat. External interference free channel access strategy dedicated to tsch. In *2018 IEEE International Conference on Electro/Information Technology (EIT)*, pages 0350–0355. IEEE, 2018.

[110] *Network simulator 3*, accessed July ,2017.

[111] Luis Pacheco, Tom Vermeulen, Sofie Pollin, and Priscila Solis. Evaluation of tsch/ieee 802.15. 4e in a domestic network environment. In *International Internet of Things Summit*, pages 257–262. Springer, 2015.

[112] Marina Petrova, Lili Wu, Petri Mahonen, and Janne Riihijarvi. Interference measurements on performance degradation between colocated ieee 802.11 g/n and ieee 802.15. 4 networks. In *Networking, 2007. ICN'07. Sixth International Conference on*, pages 93–93. IEEE, 2007.

[113] Axel Sikora and Voicu F Groza. Coexistence of ieee802. 15.4 with other systems in the 2.4 ghz-ism-band. In *Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE*, volume 3, pages 1786–1791. IEEE, 2005.

[114] Dong Yang, Youzhi Xu, and Mikael Gidlund. Coexistence of ieee802. 15.4 based networks: A survey. In *IECON 2010-36th Annual Conference on IEEE Industrial Electronics Society*, pages 2107–2113. IEEE, 2010.

[115] Sarra Hammoudi, Harous Saad, and Aliouat Zibouda. A fault tolerant and resilient infrastructure as a service for intelligent storage in iot-cloud. *Submitted to 15th International Wireless Communications Mobile Computing Conference*, 2019.

List of included publications

The thesis is based on the following papers:

A) Hammoudi, S., Benaouda, A., Harous, S., & Aliouat, Z. (2016, October). Load balancing in the Cloud using specialization. In Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE Annual (pp. 1-7). IEEE.

B) Hammoudi S., Aliouat Z., Harous S. Challenges and research directions for Internet of Things. Telecommunication Systems. 2018 Feb 1;67(2):367-385.

C) Hammoudi, S., Harous, S., Aliouat, Z., & Louail, L. (2018). Time slotted channel hopping with collision avoidance. International Journal of Ad Hoc and Ubiquitous Computing, 29(1-2), 85-102.

D) Hammoudi, S., Aliouat, Z., & Harous, S. (2018, June). A new Infrastructure as a Service for IoT-Cloud. In 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC) (pp. 786-792). IEEE.

E) Hammoudi S., Harous S., Aliouat Z. External Interference Free Channel Access Strategy Dedicated to TSCH. In 2018 IEEE International Conference on Electro/Information Technology (EIT) 2018 May 3 (pp. 0350-0355). IEEE.

F) Hammoudi S., Harous S., Aliouat Z. A Fault Tolerant and Resilient Infrastructure as a Service for Intelligent Storage in IoT-Cloud. Submitted to 15th International Wireless Communications & Mobile Computing Conference, Tangier, Morocco.

G) Hammoudi S., Aliouat Z., Harous S. Enhanced Time Slotted Channel Hopping. transactions on emerging telecommunications technologies. Submitted to Transactions on Emerging Telecommunications Technologies.

Résumé

L' émergence de l'internet des objects (IoT) a permi à un grand nombre d'appareils intelligents de se connecter à l'internet. Les réseaux de capteurs sans fil et le Cloud computing sont les principaux éléments qui facilitent l'émergence de l'IoT. La capacité de stockage et la consommation d'énergie des capteurs sont les problèmes les plus fréquents dans l'IoT. La bande passante de 2.4 GHz autorise de nombreuse technologies sans fil d'utiliser le même spectre (2.4 GHZ), ce qui entraîne un problème d'interférence très sévère. La norme IEEE 802.15.4 propose le mode TSCH (Time Slotted Channel Hopping) conçu pour les réseaux LLN (low-power and lossy networks). Le TSCH vise à améliorer la fiabilité de la transmission des données détectées en adaptant les sauts des canaux pour atténuer l'impact négatif des interférences externes. Étant donné que les signaux Wi-Fi peuvent affecter la qualité des canaux, le passage aveugle des capteurs d'un canal à un autre peut également nuire aux performances de transmission des données. Pour résoudre ce problème, nous proposons une nouvelle stratégie dédiée à TSCH qui permet aux capteurs de passer d'un canal à un autre de manière intelligente intelligente. Le TSCH utilise les liens partagés pour augmenter le débit des réseaux. Pour éviter les collisions en présence de nœuds cachés programmés dans un lien partagé, cette thèse propose deux algorithmes intelligents : Time Slotted Channel Hopping with Correct Collision Avoidance backoff algorithm(TSCH-CCA) et Enhanced Priority Channel Access Backoff Algorithm(E-PCA). Ces deux algorithmes sont appliqués respectivement aux paquets normaux et aux paquets d'événements critiques. Les solutions proposées présentent des améliorations significatives en terme de latence, de congestion du réseau, de durée de vie du réseau, du délais des paquets d'événements critiques et diminution de collisions. Nous avons également proposé deux algorithmes pour éviter les interférences externes et les évanouissements par trajets multiples : Enhanced Time Slotted Channel Hopping (E-TSCH) et Reliable Time Slotted Channel Hopping (R-TSCH). Pour évaluer les performances de nos solutions proposées,

qui surmontent les interférences externes et les évanouissements par trajets multiples, nous avons implémenté nos techniques dans Network Simulator 3 (NS3) et nous les avons comparées à TSCH. Les résultats montrent une nette amélioration en terme de nombre de paquets retransmis, de taux de distribution de paquets et de consommation d'énergie. Nous avons également proposé trois IaaSs (Infrastructure as a Service) sur le Cloud. Les trois IaaSs visent à assurer l'équilibrage de charge, à minimiser la latence des clients et à fournir un système tolérant aux pannes. Ils assurent la disponibilité des données critiques détectées par les capteurs tout en répondant rapidement aux requêtes critiques d'entrée et de sortie. Les IaaSs sont implémentées dans la platfrom JADE (java).

mots clés : L'Internet des objects, Infrastructure as a Service, Équilibrage de charge, Cloud computing, IEEE 802.15.4e, TSCH, Réseau de capteurs sans fil.

الملخص

سمحت إنترنت الأشياء (IoT) لعدد كبير من الأجهزة الذّكية بالإتّصال بالإنترنت. تعتبر شبكات الإستشعار اللاّسلكية WSNs والحوسبة السّحابية Cloud computing من العناصر الرّئيسية التي سهّلت ظهور إنترنت الأشياء. تعد السّعة التّخزينيّة وإستهلاك الطّاقة لأجهزة الإستشعار من أكثر المشكلات شيوعاً في إنترنت الأشياء. يسمح عرض النّطاق الترددي ٤.٢ جيجاهرتز للعديد من التّقنيات اللاّسلكية بإستخدام نفس الطّيف (٤.٢ جيغاهرتز) ، والذي يسبّب مشكلة تداخل شديد بين الشّبكات اللاّسلكية. يقترح معيار IEEE 802 15 4 e وضع بروتوكول TSCH المصمّم للشّبكات منخفضة القدرة و الضّعيفة (LLNs). يهدف TSCH إلى تحسين موثوقيّة إرسال البيانات عن طريق تكييف فواصل القناة للتّخفيف من الأثر السّلبي الذي يسبّبه التّداخل الخارجي. بما أن إشارات WiFi يمكن أن تؤثّر سلباً على جودة القنوات، فإن مرور أجهزة الإستشعار من قناة إلى أخرى يمكن أن يؤثّر سلباً على أداء نقل البيانات . لحل هذه المشكلة، نقترح إستراتيجية جديدة مخصّصة لTSCH التي تجعل أجهزة الإستشعار أكثر ذكاءاً وتتيح لهم القفز من قناة إلى أخرى بطريقة ذكيّة. يستخدم TSCH روابط مشتركة لزيادة سرعة نقل البيانات . لتجنّب الإصطدامات في وجود العقد المخفيّة المبرمجة في رابط مشترك ، تقترح هذه الرّسالة خوارزميتين ذكيتين : Time Slotted Channel Hopping with Correct Collision Avoidance backoff algorithm و Enhanced Priority Channel Access Backoff Algorithm . يقدّم الحل المقترح لدينا تحسينات مهمّة في زمن وصول البيانات، إزدحام الشّبكة، عمر الشّبكة، حياة حزمة الأحداث الحسّاسة، والتّخفيف من الإصطدام . لتفادي مشكل التداخل الخارجي و مشكل تلاشي الإشارات بسبب تعدّد المسيرات ، إقترحنا خوارزميتين ذكيّتين : Enhanced Time Slotted Channel Hopping و Reliable Time Slotted Channel Hopping . لتقييم أداء الحلول المقترحة قمنا بتنفيذ تقنياتنا في NS3 و تمّت مقارنتها مع TSCH . تظهر النّتائج تحسينات كبيرة في عدد الحزم المعاد إرسالها ، ومعدّلات تسليم الحزم ، و إستهلاك الطّاقة. فيما يخص الحوسبة السّحابية، إقترحنا ثلاثة بنيات أساسيّة كخدمة على السّحابة. تم تصميم البنيات الأساسيّة لتوفير موازنة الحمل، وتقليل وقت إستجابة العميل وتوفير نظام التّسامح مع الأخطاء. هذه البنيات تضمن توافر البيانات الهامّة التي تلتقطها أجهزة الإستشعار مع الإستجابة السّريعة لمطالب التّخزين و التّحميل. تم تطبيق هذه البنيات على اللّوح جاد JADE .

الكلمات الرّئيسية: إنترنت الأشياء، شبكات الإستشعار اللاّسلكية، الحوسبة السّحابية، TSCH ، IEEE 802.15.4e