ROYAUME DU MAROC
Université Mohammed V - RABAT -
جامعة محمد الخامس – الرباط –
Faculté des sciences
كلية العلوم
CENTRE D'ETUDES DOCTORALES - SCIENCES ET TECHNOLOGIES

N° d'ordre : 3455

THESE

En vue de l'obtention du : ***DOCTORAT***

Structure de Recherche : Laboratoire de Recherche en Informatique et Télécommunications
Discipline : Sciences de l'ingénieur
Spécialité : Informatique et Télécommunications

Présentée et soutenue le : 17/04/2021 par :

Mohammed ASSAOUY

Artificial Intelligence-based Energy Optimization : Applications in Internet of Things Communications

JURY

Mohammed RZIZA	*PES, Université Mohammed V-RABAT-, Faculté des Sciences, Rabat.*	*Président*
Mohamed OUADOU	*PES, Université Mohammed V-RABAT-, Faculté des Sciences, Rabat.*	*Directeur de thèse*
Ouadoudi ZYTOUNE	*PES, Université Ibn Tofail, ENSA, Kénitra.*	*Co-directeur de thèse*
Khalid MINAOUI	*PES, Université Mohammed V-RABAT-, Faculté des Sciences, Rabat.*	*Rapporteur/Examinateur*
Abdelilah JILBAB	*PH, Université Mohammed V-RABAT-, ENSAM, Rabat.*	*Rapporteur/Examinateur*
Mohamed EL AROUSSI	*PH, EHTP, Casablanca.*	*Rapporteur/Examinateur*

Année Universitaire : 2020/2021

Avant-Propos

C'est avec un immense plaisir que je consacre ces quelques lignes en signe de reconnaissance et de gratitude à toutes les personnes qui ont contribué directement ou indirectement à la concrétisation de ce travail de thèse de Doctorat. Je vous remercie tous et toutes du fond du coeur.

Les travaux présentés dans ce mémoire de thèse ont été conduits au sein du Laboratoire de Recherche en Informatique et Télécommunications (*LRIT*) à la Faculté des Sciences de Rabat-Université Mohammed V, sous la direction de Monsieur le Professeur Mohamed OUADOU et l'encadrement de Monsieur le Professeur Oudoudi ZYTOUNE.

La direction des travaux de cette thèse a été d'abord assurée par Feu le Professeur Driss ABOUTAJDINE, que Dieu ait son âme en sa sainte mésiricorde, puis par Professeur Mohamed OUADOU qui m'a fait l'honneur de bien vouloir me permettre de continuer cette entreprise sous sa direction et pour m'avoir donné la chance de poursuivre ce travail de recherche sur ce projet passionnant au sein de son équipe de recherche.

Je tiens à remercier en haut lieu Monsieur Mohamed OUADOU, Professeur d'enseignement supérieur à la Faculté des sciences de Rabat et directeur de la structure de recherche LRIT pour ses encouragements, sa disponibilité, ses idées, ses conseils et sa sympathie qui m'ont permis de mener à bon port les travaux de cette thèse.

Je tiens à remercier sincèrement et chaleureusement mon encadrant de recherche, Monsieur le Professeur Ouadoudi ZYTOUNE, pour m'avoir beaucoup aidé tout au long de mon parcours de recherche. Je tiens à lui exprimer ma gratitude pour ses conseils con-

tinus, son immense savoir et son attitude de recherche positive qui m'ont été fournis au cours de toutes ces années de recherches menées dans le cadre de cette thèse. Ce projet n'aurait jamais été accompli sans son soutien et ses précieux conseils.
Mes scincères remerciments au Président du jury Professeur Mohammed RZIZA ainsi qu'aux Professeur Khalid MINAOUI, Professeur Abdelilah JILBAB et Professeur Mohamed EL AROUSSI pour avoir accepté d'investir de leur temps pour l'évaluation de mon mémoire de thèse.
Je voudrais aussi remercier mes collègues et amis du Laboratoir LRIT, en particulier, mes anciens du cycle doctoral, Mourad, Asmae, Mohamed et Aziz pour ces échanges agréables qu'ont avait toujours eu ensemble et j'avoue que pour moi ces échanges me fournissaient plus de motivation à aller de l'avant même dans les moments les plus difficiles.
J'adresse également mes remerciements aux directeurs qui se sont succédés à la tête du CEDOC, à son staff administratifs ainsi qu'aux techniciens de la structure de recherche doctorale du laboratoire LRIT pour leurs dévouements, respects mutuels et sympathies.
Un grand merci à mon père et à ma mère. Ils m'encouragent toujours avec leurs meilleurs voeux. Et je suis éternellement reconnaissant à mes parents aimants pour leur soutien incroyable, leur amour sans fin, et leur patience tout au long de ma vie. Ils ont fait des sacrifices pour moi, que je ne pourrais jamais rembourser.
Mes remerciements vont aussi à tous mes frères et surtout mes adorables soeurs, Amal et Samia pour leur encouragements indefectibles ainsi qu'à mes splendides nièces Malak et Inès et à tous mes adorables neveux.

Abstract

Most common power source for connected objects still mainly battery-based. With limited power capacity, objects continually require local intervention to replace batteries when empty. Which can be difficult to achieve, especially when placed in hostile and dangerous environments.

This thesis aims at proposing communication strategies, allowing the performance optimization of connected objects when using a single battery. Both primary and secondary batteries have been considered. For primaries, we have exploited the battery recovery effect. For secondaries, batteries were assumed to be equipped with energy harvesters.

Three scenarios have been considered depending on the information being available. In the first case, it was assumed that stochastic information was available at the beginning of the optimization process. In the second case, the information was considered unavailable and the system completely unknown which has required a significant amount of time to learn optimal policies. In the third case, it was assumed that the information was partially known. This partial knowledge has significantly reduced the calculation time for a performance close to that of the first scenario.

Keywords: Internet of Things; Artificial intelligence; Recovery effect; Energy harvesting; Dynamic programming; Reinforcement learning.

Résumé

La source d'énergie la plus courante pour les objets connectés est encore principalement basée sur les batteries. Avec une capacité limitée en energie, ces objets nécessitent continuellement une intervention locale pour remplacer les batteries lorsqu'elles sont vides. Ce qui peut être difficile à réaliser, surtout lorsqu'ils sont placés dans des environnements hostiles et dangereux.

Cette thèse vise à proposer des stratégies de communication permettant d'optimiser les performances des objets connectés utilisant une seule batterie. Les cas de batteries primaires et secondaires ont été pris en compte. Pour les batteries primaires, l'effet de recouvrement a été exploité. Pour les batteries secondaires, la recharge au moyen de récolteurs d'énergie est considérée.

Trois scénarios ont été envisagés en fonction des informations disponibles. Dans le premier cas, il a été supposé que les informations stochastiques sur le système étaient disponibles au début du processus d'optimisation. Dans le second cas, ces informations ont été considérées comme non disponibles et le système supposé totalement inconnu ce qui a nécessité un temps d'apprentissage significatif des politiques optimales. Dans le troisième cas, les informations étaient supposée partiellement connues. Cette connaissance partielle a permis de réduire significativement le temps de calcul pour une performance proche de celle du premier scénario.

Mots clés: Internet des Objets; Intelligence Artficielle; Effect de recouvrement; Récolte d'énegie; Programmation dynamique; Apprentissage par renforcement.

PUBLICATIONS

- M. ASSAOUY, O. ZYTOUNE and M. OUADOU, "Rapid learning optimization approach for battery recovery-aware embedded system communications", International journal of communication networks and information security, 12, no.3, 2020.
- M. ASSAOUY, O. ZYTOUNE and M. OUADOU, "Battery Recovery-Aware Optimization for Embedded System Communications". Wireless Personal Communications, 110(4), 1929-1946, 2020.
- M. ASSAOUY, O. ZYTOUNE and D. ABOUTAJDINE, "Policy iteration vs Q-Sarsa approach optimization for embedded system communications with energy harvesting", In : 2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). IEEE, p. 1-6, 2017.
- M. ASSAOUY, O. ZYTOUNE and D. ABOUTAJDINE, "DP and RL Approach Optimization for Embedded System Communications with Energy Harvesting", I4CS 2017, CCIS 717, pp. 167-182, 2017.

Table of Contents

Acronyms

6LoWPAN	IPv6 Low power Wireless Personal Area Networks
ASIC	Application Specific Integrated Circuit
ADC	Analog to Digital Converter
AI	Artificial Intelligence
AIoT	Artificial Intelligence of Things
AmI	Ambiance Intelligence
ATM	Asynchronous Transfer Mode
CASAGRA	Coordination And Support Action for Global RFID-related Activities
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DP	Dynamic Programming
DSSC	Dye-sensitized Solar Cell
IBSG	Internet Business Solution Group (Cisco)
ICT	Information and Communication Technology
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IIS	Institute of Integrated Circuits
IoT	Internet of Things
IPSO	IP for Smart Objects
IPv6	Internet Protocol version 6

LAN	Local Area Network
M2M	Machine to Machine
MAC	Medium Access Control
MDP	Markov Decision Process
MCU	Microcontroller Unit
MEMS	Micro-electromechanical systems
ML	Machine Learning
MP	Markov Process
MRP	Markov Reward Process
PA	Power Amplifier
PHY	Physical layer
PI	Policy Iteration
PS	Power Supply
RF	Radio Frequency
RFID	Radio Frequency Identification
ROLL	Routing Over Low power and Lossy networks
RPL	Routing Protocol for Low-Power and Lossy Networks
RL	Reinforcement Learning
RLA	Rapid Learning Algorithm
SoC	State of Charge
SoH	State of Health
TCP/IP	Transmission Control Protocol/Internet Protocol
Ubicomp	Ubiquitous computing
UDP	User Datagram Protocol
ULV	Ulta Low Voltage
VI	Value Iteration
VM	Voltage Multiplication
WSN	Wireless Sensor Network

List of Figures

General Introduction

Context of the study

The Internet of Things (IoT) technology solution is in the majority of cases battery-powered, and therefore its performance is directly related to the ability of the batteries to maintain the life of the device or monitoring system and to the optimal use of these batteries. However, the lifetime of IoT devices may be short due to the limited charge of the batteries, which therefore need to be replaced frequently, unless an appropriate solution is found to extend the lifetime of the batteries in question. The challenge is then to provide solutions to mitigate the maintenance costs resulting from battery replacement, maintenance visits associated with the difficulty to access thousands or millions of IoT devices such as wireless sensor nodes located mostly in hostile or difficult to access terrain.

Consequently, optimizations and energy saving plans must be proposed to extend the maximum life of the batteries to the best of their capacities. This is the main reason why the energy optimization of wireless sensor terminals spread over several sectors has become a concern and a significant amount of works is published every year on this subject.

In fact, IoT devices are massively deployed in particular to detect the environment, measure and send wireless data to different control units for processing and subsequent decisions. IoT-based systems form the basis for applications covering various areas of monitoring for multiple purposes such as early detection of disasters like forest fires [1, 2], floods

[3], monitoring of river pollution [4] or even monitoring of large structures like bridges [5] or buildings [6] and for the understanding of natural phenomena such as volcanic eruptions [7] and landslides [8], etc.

The objective of the work presented in this thesis is to implement an intelligent capability to wireless sensor nodes, as IoT devices, in order to significantly improve their autonomous behavior and functioning. In the context of peer-to-peer communication, nodes will be able to take into account changes dictated by the characteristics of channel states and the size of data packets, and adapt accordingly to changing conditions. Depending on the type of batteries used, whether primary or secondary, the objective is to extend the lifetime of an embedded system and maximize the amount of data produced over a limited period of time when only one battery is used. The implementation of artificial intelligence capabilities based on machine learning in wireless sensor nodes is exploited in this thesis to achieve these goals.

Problem statement

Today, wireless communications are of great importance, particularly in the area of the Internet of Things, where a wide range of devices, sensors and machines can exchange data and make decisions on behalf of people. When such elements are spread over large areas, great challenges must be considered. In the case of wireless sensor nodes, one of the main challenges is energy limitation. In fact, a wide range of research groups are focusing on the energy management of connected objects in order to improve their energy consumption efficiency. In the specific case of embedded systems and wireless sensor networks, batteries remain the main power source and, as in many cases, devices such as sensors are spread over large, inaccessible fields where replacing empty batteries is quite impossible. Furthermore, when the batteries are empty, the devices or sensors become inoperative and are therefore reported lost.

In our work, we consider both primary and secondary batteries. For primary ones, the internal battery phenomenon we have exploited, which has been described so far in this thesis, is the recovery effect. With secondary batteries, we have exploited the case of

energy harvesters implemented to the distributed systems which collect energy from the surrounding environment.

For the objective of efficient power management in embedded systems equipped with non-rechargeable batteries, many previous works have proposed different solutions exploiting the features either of the MAC layer and the physical layer of the network stack. In the first case, the transmission protocols of the MAC layers are improved to extend battery life and maximize the volumes of data transmitted [9, 10]. In the second case, low-power wireless sensor networks have been proposed based on optimized hardware designs [11]. Other work has focused on the optimization of wireless sensor networks equipped powered by batteries with harvesting capabilities as in [12, 13].

Three types of problems are addressed in this thesis, depending on the information availability. In the first case, the problem of decision making is considered when sufficient information about the system and its environment is available in both cases of batteries. In the second case, as in real cases, this information may not be available, and the sensor node must discover the environment in which it operates itself and therefore make the appropriate transmission decisions in order to maximize the volume of data transmitted and battery life. In the third type of problems, it is assumed that only a part of the information needs to be completed by a learning method and the treatment of this type of problem becomes a mixture of the two previous cases with regard to the learning and optimization processes. In all these cases, the data transmitted packets shall be considered valuable and timely when they take place during a period of time fixed in advance on the basis of a given discount factor. Otherwise, the data packets are assumed to be useless.

In this thesis, we also propose algorithms based on machine learning for peer-to-peer communications of IoT devices. The measures used in this work are the quantity of data being transmitted over a fixed period of time and the lifetime of the system when a single battery is used. Therefore, the best performing algorithm is the one that produces the largest amount of data that has been successfully transmitted, combined with a long lifetime.

Performance evaluation

In order to evaluate the performance of an IoT device in peer-to-peer communication context, the volume of data transmitted will be considered as the main key performance indicator during the lifetime of the embedded system. For sensor nodes as IoT devices, this means that the useful information is that which would be transmitted over the communication channel from the moment of deployment of the sensor node until its energy depletion or the end of the transmission period according to a discount factor. The battery life is also crucial, in relation to the activation period, to ensure continuous data transmission activity between the connected objects during a predetermined period of use. Both cases of secondary batteries with energy harvesting capacity and primary batteries with recovery effect are taken into account to improve data transmission volumes while ensuring long battery lifetime. It is obvious that a large amount of transmitted data and a long service life are linked, because an IoT device will only transmit more data packets if its battery still has enough power to supply it.

The performance obtained is closely linked to the prior or acquired knowledge of the data collected as characteristics of the data transmission environment. In realistic cases, information on the system environment is not always available in advance. It may therefore be useful to exploit algorithms from the field of machine learning, e.g. reinforcement learning algorithms, to cope with such a deep lack of information. In fact, reinforcement learning will allow sensor nodes to learn the characteristics of the environment before designing transmission decision policies.

Performance evaluation is performed for many situations concerning the availability of sufficient or no knowledge about the system environment, both in the case of batteries with harvesting capacity (i.e. secondary batteries) and recovery-aware batteries (i.e. primary batteries) used as energy sources for the embedded system under consideration.

Thesis outline

The thesis is organized as follows:

- Chapter 1 as titled "IoT communication enhancement using artificial intelligence" reviews the domain of the Internet of Things and the concepts and tools related to artificial intelligence that can be applied to IoT devices to improve their performance.

- Chapter 2 which is titled "Literature review on batteries for embedded systems" deals with the description and modeling of batteries. Secondary (rechargeable) batteries that have energy harvesting capabilities and primary (non-rechargeable) batteries that exploit the recovery effect are presented and discussed.

- Chapter 3 that is titled "Reinforcement learning: A literature review" focuses on the introduction of the Markov decision-making processes (MDP) concepts and tools and lists some solutions to these types of problems using techniques from the fields of Dynamic Programming (DP) and Reinforcement Learning (RL).

- Chapter 4 as titled "Battery-powered embedded IoT systems with energy harvesting capabilities" presents two of the published scientific contributions dealing with the energy optimization of embedded systems with energy harvesting. In the particular case of batteries with energy harvesting capacity, the performance of embedded systems in peer-to-peer communications can be greatly improved by using algorithms both from the fields of dynamic programming and reinforcement learning.

- In Chapter 5 with the title "Embedded IoT systems with recovery-aware Batteries" the interest goes to the use of primary batteries in embedded system communications. Based on the recovery effect, the use of the battery is optimized both by increasing its lifetime and the amount of data transmitted. A new rapid learning approach is proposed, which aims to use the partial knowledge available on the communication environment to enable the renewal of the internal charge of the batteries through the recovery effect.

- Chapter 6: Conclusion and future work

  This chapter concludes the thesis by presenting the conclusions and future directions of the research.

Chapter

1 IoT communication enhancement using artificial intelligence

Today, billions of objects and humans are connected through the Internet. In 2020, the connected objects were likely more than 50 billion connected devices. Therefore Internet of Things (IoT), also known as the Internet of objects, is considered a powerful tool for creating, modifying and sharing an unlimited amount of information. In fact, IoT-based information communication technologies have revolutionized the sharing of information between devices and between devices and humans, i.e. smart things like sensor nodes will be able to connect, transfer data and make their own decision on behalf of people. The operation of IoT relies mainly on sensor nodes and connected other objects placed in various sites to capture different types of data and transmit to final destinations using wireless network communications.

The fields of IoT applications are growing especially in the areas of energy management, industry, agriculture, security, transport or even health. So when IoT is combined with artificial intelligence, operations of connected objects becomes more efficient, interactions with humans are improved and data management and analysis are enhanced.

This chapter will be devoted first to the presentation of both IoT and Artificial intelligence concepts and tools and to the combination between artificial intelligence and IoT devices, such as wireless sensor nodes, (known as Artificial Intelligence of Things) to design optimal solutions especially in constrained optimization contexts.

1.1 Internet of Things

1.1.1 Internet of Things definition

The Internet has grown exponentially over the past four decades, moving from a private network of few nodes to a global public network of billions of nodes. Almost 20 years ago, the Internet of Things (IoT) paradigm was invented by Kevin Ashton in 1999 [14], and the expression of Internet of Things as stated by the author refers to the trend towards interconnected things which collect data by detecting and performing calculating on the sensor data.

Recently, The Internet of Things (IoT) has become a subject of great interest to the research, scientific and technological communities. While wireless sensors have been around for more than a few decades, the advent of smart wireless sensor networks has opened up a whole new fields of IoT applications. IoT includes Internet-connected devices, smart connected devices, wireless sensor networks, machines and devices that communicate wirelessly, ubiquitous computing (ubicomp), ambiance intelligence (AmI), and smart matter (micro-electromechanical systems (MEMS)).

Actually, the Internet of Things is different from traditional wireless sensor networks as well as computer networks and therefore poses more challenges to be solved on issues such as limited energy, restricted lifetime, etc. It is the next revolution after the great success of the Internet, which has brought a new dimension to the world of information and communication technologies (ICTs) [15].

In fact, advances in sensor technology, intelligent instrumentation, wireless sensor networks, miniaturization and information processing are contributing to the realization of the Internet of Things. For example, Data collection solutions based on wireless sensor networks (WSNs) were imagined early in the IoT paradigm because of their important features such as low-cost, long-term versatile sensing and actuation capabilities and resilient and distributed bidirectional communications capabilities. IoT is a large term that doesn't have a single definition but it includes disparate and various objects as "things" ranging from banal physical things to complex sensors as bio-sensors. IoT things have the capability to (1) sens/actuate (2) connect to the internet and (3) process data [16].

In the literature, the concept of the Internet of Things (IoT) is defined in different ways, although they are quite similar. The US National Intelligence Council provides a definition currently in use, and maybe simpler to apprehend [17] as follows :"The IoT is the general idea of things, especially everyday objects, which are readable, recognizable, locatable, addressable, and controllable via the Internet whether via RFID, wireless LAN, wide-area networks, or other means". According to the Cisco Internet Business Solutions Group (IBSG), "IoT is simply the point in time when more things or objects were connected to the Internet than people" [18].

However, the definition given by the CASAGRAS consortium (Coordination And Support Action for Global RFID-related Activities And Standardization, Grant agreement ID: 216803) is perhaps the most general as it refers both to a Things-oriented and an Internet-oriented visions by defining the IoT as follows: "IoT is a global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor and connection capability as the basis for the development of independent cooperative services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity and interoperability" [19].

1.1.1.1 The temporal path of the evolution of the IoT

IoT has evolved over a long way that can be divided into four phases, as given in Fig. 1.1. In phase one, IoT was materialized by the emergence of bar codes and radio-frequency identification (RFID) that have simplified the inventory system management many industries and provided them with some form of identification and tracking. Phase two involved the connectivity of the WSN to the Internet by means of gateways. WSN has large capabilities of sensing and actuating that provide excellent support for IoT applications [20, 21, 22] and has widely opened the emergence of new civil and military applications. The third phase was marked by the emergence of new generation of IoT applications that enable big data analysis. Finally, the fourth phase will lead to build bridges between the different sectors of life (healthcare, energy, industry, transportation, military, retail and

buildings) to make them connected and allow them to interact with each other. Many WSNs are not based on Internet Protocol (IP) such as Zigbee which limit its connectivity to the external world [22]. Nowadays, the trend is towards the generalization of IP connectivity between WSN and Internet to make IoTs [23].

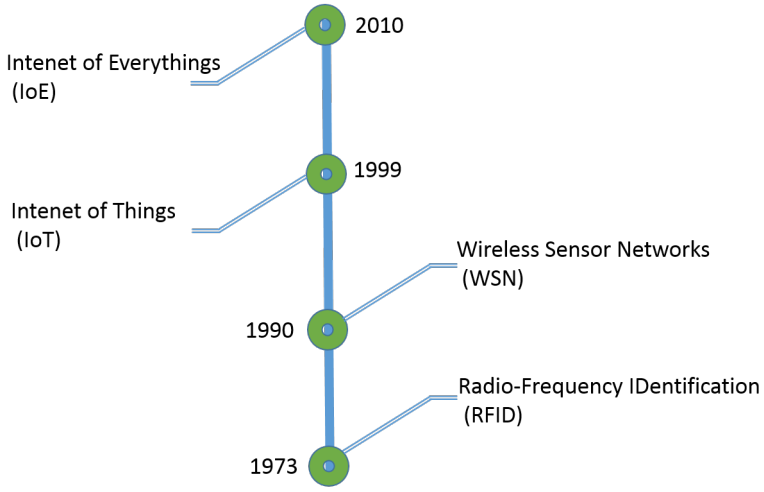

Figure 1.1: Evolution of IoT

1.1.1.2 Market place evolution for IoT devices

The vision of the IoT has been heavily powered by statistics and predictions. In this section, we discuss some statistics and facts related to the IoT which allows us to understand how the IoT has grown over the years and how it is expected to grow in the future. Further, the statistics and facts in Fig. 1.2 highlight the future trends in the industry marketplace and in Fig. 1.3 we present a chart showing the Google trends of growing interest in Internet of Things for all categories. By the end of 2020, it is estimated that there will be 50 to 100 billion devices connected to the Internet, ranging from smartphones, PCs, and ATMs (Automated Teller Machine) to manufacturing equipment in factories and products in shipping containers [24]. As depicted in Fig. 5.1 the number of things connected to the Internet exceeded the number of people on Earth in 2008 and, according to CISCO, each individual on earth will have more than six devices connected to the Internet by 2020.

The interconnection and communication between everyday objects, in the IoT paradigm,

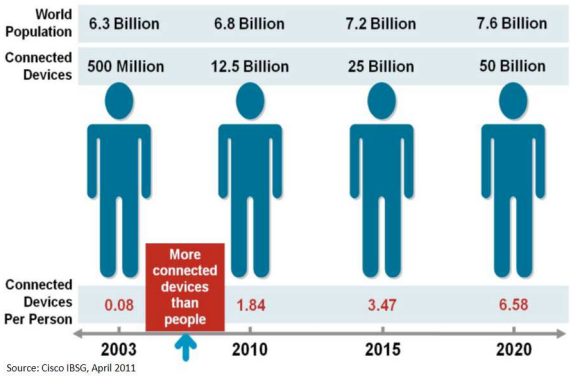

Figure 1.2: Internet of Things market place evolution

Figure 1.3: Interest in IoT over time through Worldwide Web Search from 2004 till March 2020, source: Google trends

enables many applications in variety of network domains as depicted in Fig. 1.4. As IoT evolves, these networks, and many others, will be interconnected and provided with additional security, analysis and management capabilities as shown in Fig. 1.5. This will allow IoT to become even more powerful in what it can help people achieve.

1.1.2 IoT versus Wireless Sensor Networks

The IoT devices use the Internet Protocol (IPv6) to ensure that every "thing" has an identifiable Internet address [25], where the WSN only has the option of using the Internet, without this being required for it. Nevertheless, the WSN only reaches its full potential in terms of performance when it is connected to the Internet, thus forming part of the IoT [26]. Actually, WSN technology becomes an essential component of IoT because it is a set of sensor nodes connected by wireless channels, and, as is the case in IoT, specific

Figure 1.4: Main application domains of IoT devices

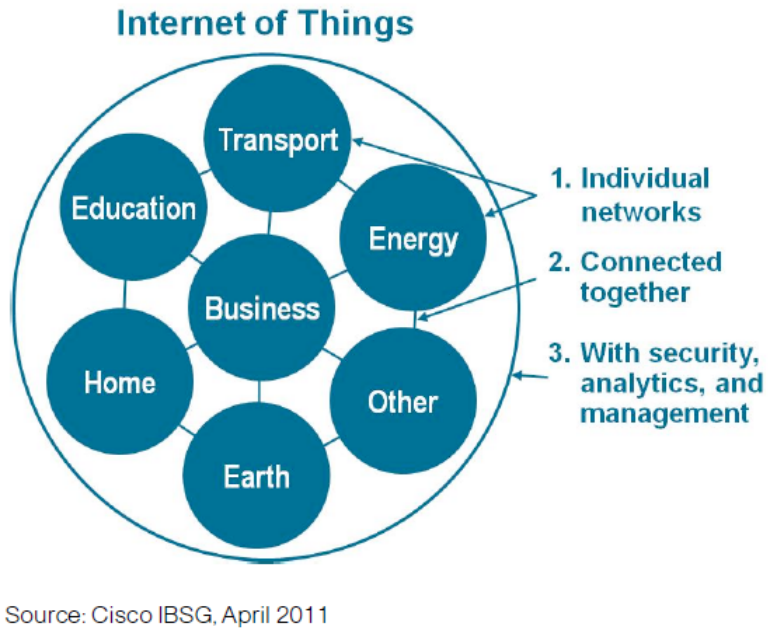

Figure 1.5: IoT as a Network of Networks

applications of WSN can be applied to a wide variety of fields such as health, agriculture, logistics and others. In addition, recent works considered as WSN applications are also called IoT applications without distinction between the specific characteristics of each of these denominations as in [26].

1.1.2.1 The IoT protocol stack standards

Standards for IoT protocol stack are developed by the Institute of Electrical and Electronics Engineers (IEEE) and the Internet Engineering Task Force (IETF) [27]. A representative view of an IoT protocol stack is given in Fig. 1.6.

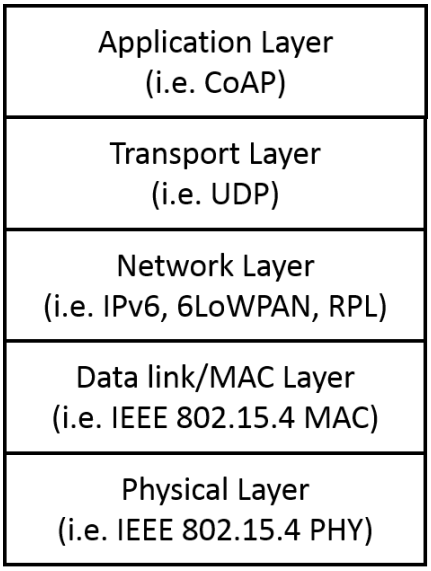

Figure 1.6: Representative view of an IoT protocol stack

Application Layer: At this layer the application and data are presented in a visual form that users can understand. CoAP is the most common protocol being used for IoT applications. It relies on UDP for extending the use of web communication services to IoT applications including M2M applications. It is designed specifically for IoTs that are operating in environments where strict constraints are encountered in terms of limited energy resources, limited computation capabilities and high packet loss rates. CoAP promises to become the ubiquitous IoT application layer protocol in the future [28, 29].

Transport Layer: This layer provides applications with communication services such as reliability, flow control, and multiplexing. For connectionless applications cases, the transport Layer protocol is mainly UDP. Since this protocol simply sends packets, its bandwidth overhead and latency are much lower compared to the case of Transmission

Control Protocol (TCP). UDP also provides checksum to verify that data has arrived intact by adding an error detection flag.

Network Layer: This layer is in charge of packet forwarding and routing through routers in the neighboring. Making use of IP is necessary for communication through Internet. In fact, IP for Smart Objects Alliance (IPSO) actively promotes the use of IP-embedded devices for machine-to-machine applications. In fact, IPv6 is necessary since it offer an address space of 128 bits that fits with the unlimited growing number of connected objects as stated earlier. With IPv6 3.4×10^{38} addresses are then allowed.
While trying to reshape the world of IoT, the IETF IPv6 over Low power WPAN (6LoWPAN) working group has started to work on specifications for transmitting IPv6 over IEEE 802.15.4 networks since 2007. 6LoWPAN adapts link layers to support IPv6 packets over IEEE 802.15.4 frames. Typically, small packet sizes , low bandwidth, battery supplied devices, low cost, large number of devices, unknown node positions and high unreliability are the main characteristics of Low power WPANs [30]. 6LoWPAN combines Internet Protocol (IPv6) and Low-power Wireless Personal Area Networks(LoWPAN). Therefore, it allows objects with limited processing capacity to transmit information over wireless channel using IP. It is ZigBee's newest competitor. 6loWPAN can communicate with IEEE802.15.4 devices and other type of devices on IP network link such as Wifi.
For 6LoWPAN, routing problems are the most challenging issues. Solutions that take into account the 6LoWPAN mechanisms and IPv6 behavior along with the specific application requirements will be succesful [31]. The IETF "Routing Over Low power and Lossy (ROLL) networks" working group is developing an effective solution. Recently, an IPv6 Routing Protocol for Low-power and Lossy Networks (RPL) [32] has been proposed by this group. Actually, RPL is a network layer protocol for low power consumption wireless networks. It works on IEEE 802.15.4 and supports M2M communications in various link layers, especially those implemented in devices with limited resources.

Data link/MAC Layer: The data link layer provides the procedural and functional tools to transfer data between network nodes and might provide the means to detect and correct errors that may occur in the physical layer. Data link controls determine who

should send the data, how much data should be sent and how errors can be detected and corrected. IEEE 802.15.4 defines the link and MAC protocols. In MAC protocol, the layer interacts directly with the radio and defines the MAC header format (with fields such as source and destination address) and how nodes can communicate with each other.

Physical Layer: This layer determines the physical radio operation frequency, the signal modulation and encoding. Low-power PHY layer and hardware together with an energy-efficient MAC layer are important for connectivity in case of smart objects. IEEE 802.15.4 is the most important standard in low-power radio technology [33]. It defines both the MAC layer (e.g., which node should talk and when, which channel to use) and PHY layer (e.g., the signal modulation to be used). The IEEE 802.15.4 working group published the first revision of the standard in 2003 and a second revision in 2006. Several working groups (e.g. IEEE 802.15.4e) are currently involved in enhancing the standard and preparing its next revision. IEEE 802.15.4 for PHY layer makes trade-off between data rate, energy-efficiency and range in medium-sized networks. Several PHY layers are defined by the standard, but the most used physical frequency channel is that of $2.4-2.485$ GHz band, which is a worldwide unlicensed band.

1.1.2.2 The IoT challenges

Before we dive into the Artificial Intelligence powered Internet of Things, we will describe some of the challenges that the IoT still has yet to overcome [34], such as:

- Energy management of sensors, although there have been large improvements in this aspect with the appearance of technologies that allow to recharge batteries without the need of an external power source such as producing energy through bodily movements and through the usage of environmental energies. Improvements in this area are even more evident in the new paradigm since these objects need to have computing capabilities to enable them to take decisions and act accordingly;
- Connectivity: the still on going deployment of IPv6, which has not reached most end users, and the fact that the range of available IPv4 addresses is clearly insufficient. In addition, the possibility of having billions of things uniquely addressable

over the Internet, presents a challenge to the goal of ubiquitous computing, which IPv6 should solve with the advantages of allowing easier network management and improved security features;

- Creation and usage of standards for the various systems that are connected by this paradigm. This will enable IoT to evolve so that networks and various sensors become integrated and standardized.

IoT enables a state of ubiquitous networking in which every object in our daily lives could be a potential connected thing on the internet, with its own set of sensors, actuators and, possibly, these could even have a sense of purpose and intelligence to react to their surroundings, which leads us to the Artificial Intelligence.

1.2 Artficial Intelligence and machine learning

1.2.1 Definition

Artificial intelligence (AI) is the intelligence that can manifested by machines or software, it is also the academic name for the area of study that focuses on how to make computers and software behave intelligently. In fact, John McCarty was the first scientist to invent the term Artificial intelligence in 1955 [35]. He defines AI as the science and engineering of making intelligent machines. Some of the fields where Artificial Intelligence can be applied are, for example, video games, data mining, discourse recognition, natural language understanding, Computer Vision, the creation of expert systems and data classification [36]. Thus, Artificial Intelligence can be divided into several branches among which stand out Computer Vision, Fuzzy Logic, Natural Language Processing, Heuristics and Machine learning.

1.2.2 Machine learning

Machine Learning, is one of the branches of Artificial Intelligence, and, maybe, one of the most important subfields of AI [37]. It is based on the development of learning

techniques to allow machines to learn [38], based on the analysis of data and applying some algorithms for this processing and others for decision making from the data already analyzed. According to Gartner's hype cycle for emerging technologies, machine learning is at the top of the plot, precisely in the peak area of inflated expectations, as shown in the small box on the right in Fig. 1.7 [39], which means that it raises high expectations. Currently, ML is used in many areas and industries to make improvements and find innovative solutions to business problems. For example, American Express uses machine learning algorithms to analyze data and detect fraud in near real time to prevent millions of losses. And Volvo uses data to help predict when parts may fail or when vehicles need to be repaired to improve vehicle safety.

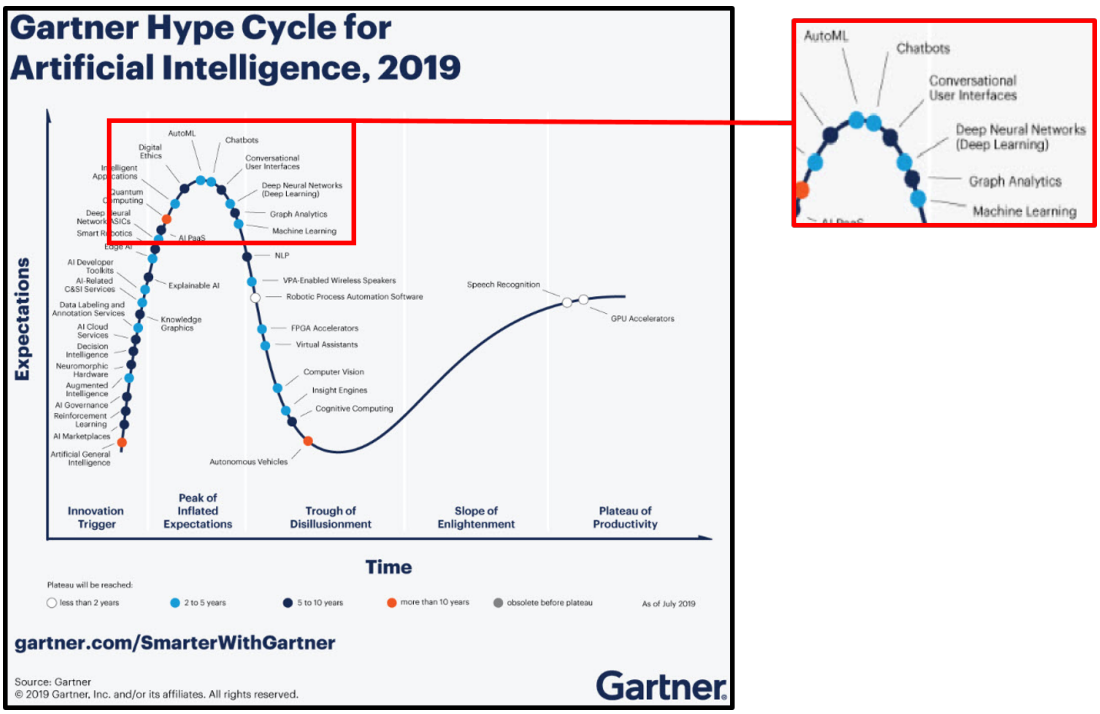

Figure 1.7: Gartner Hype Cycle for Emerging Technologies for 2019

Machine Learning algorithms can be classified based on the type of feedback received. In this subsection, three main categories of machine learning: supervised learning, unsupervised learning and reinforcement learning as given in Fig. 1.8.

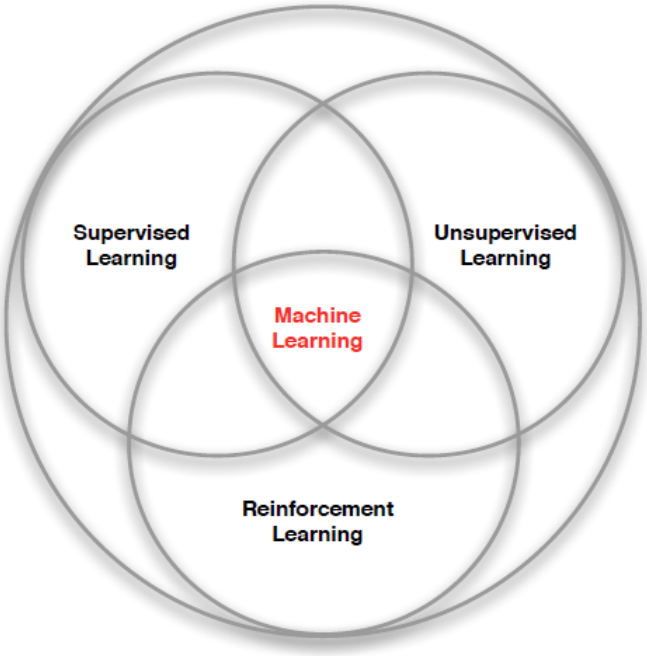

Figure 1.8: Machine learning categories

1.2.2.1 supervised learning

These algorithms infer a function by means of labeled training data that they receive as input [40, 41, 42] . An example of them is the algorithms used to classify images by using Computer Vision techniques as in Fig. 1.9. These algorithms must be fed with images labelled, for example, as correct or incorrect. In this way, the algorithm is fed so that it learns to classify the images according to the previously labeled images supplied. Therefore, it can be said that the algorithm learns for itself [43]. Some of the algorithms found in this group are some of those belonging to Artificial Neural Networks, Bayesian Networks, Support Vectors Machines and Decision Trees.

1.2.2.2 Unsupervised learning

The difference with the supervised learning case is that the input data is not labeled, and it is the system itself that must infer algorithms or patterns to recognize and label the same type of data as in Fig. 1.10. In this category [42, 43], we find some types of Artificial Neural Networks, the Association Rules and some Grouping Algorithms. The combination of the two previous algorithms can be performed [44] as a semi-supervised case. In this

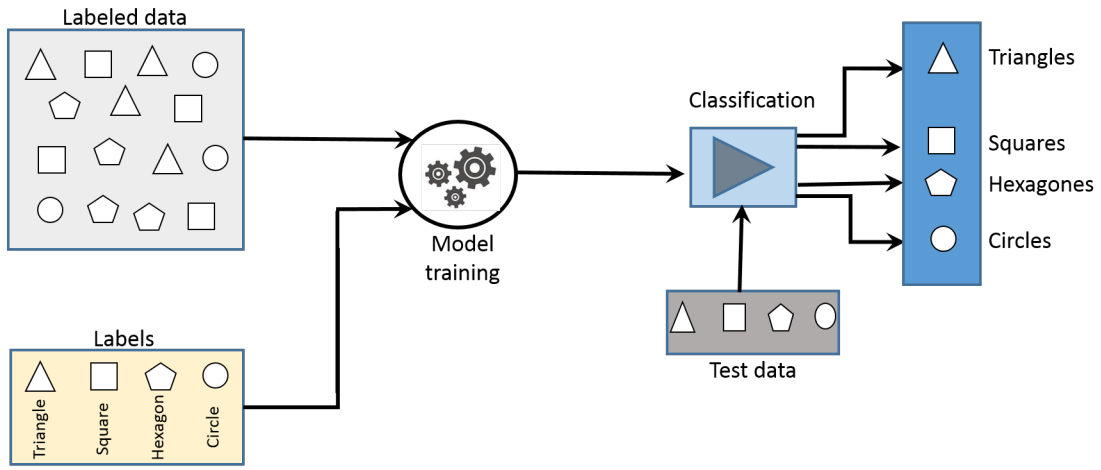

Figure 1.9: Supervised learning

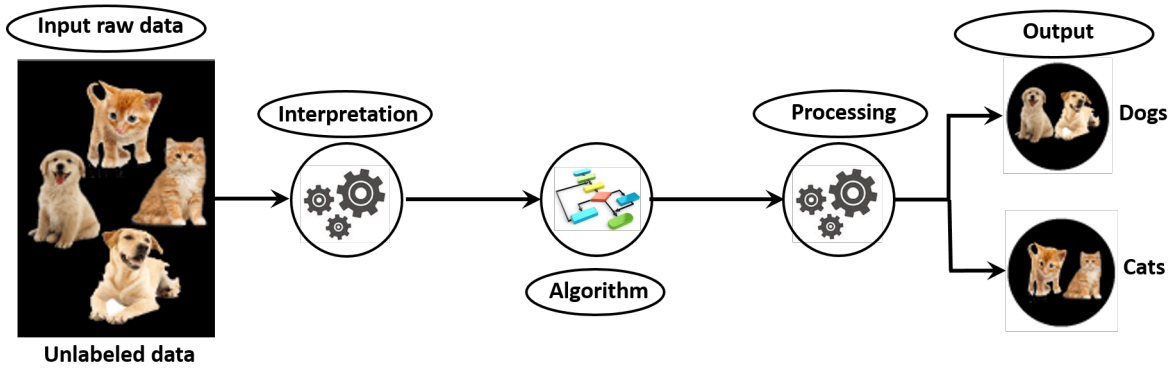

Figure 1.10: Unsupervised learning

situation, the system must consider both the tagged elements and the untagged elements.

1.2.2.3 Reinforcement learning

Reinforcement learning [45] consists in learning what to do and how to match particular states or situations to actions, so as to maximize a possible reward. The agent as learner is not informed of the actions to be taken in each situation, as it is the case in most forms of machine learning, he must then find out which action brings the most reward by trying all of them. In the most interesting and difficult cases, actions can affect not only the immediate reward but also the following situation and, through this, all the following rewards. The two main stages that of trial and error research and delayed reward are the most important indicators that distinguish the characteristics of reinforcement

learning. Reinforcement learning is different from supervised learning, the kind of learning studied in most current research on machine learning. Supervised learning is learning from examples provided by a knowledgeable external supervisor. This is an important kind of learning, but alone it is not adequate for learning from interaction. In interactive problems it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. In uncharted territory -where one would expect learning to be most beneficial- an agent must be able to learn from its own experience as ilustrated in Fig. 1.11.

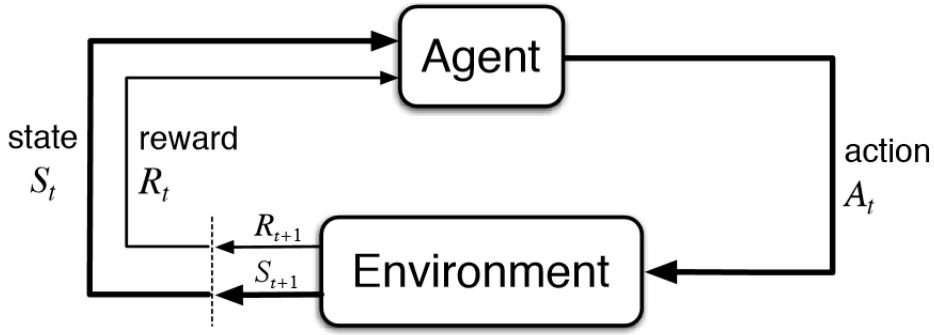

Figure 1.11: Reinforcement learning structure

1.3 Artificial Intelligence of Things

Artificial Intelligence of Things (AIoT) combines Artificial Intelligence technologies with the infrastructure of the Internet of Things. As a result, the functioning of connected objects is more efficient, human-computer interactions are improved, and data management and analysis are more advanced. Thus, the main objective of AIoT is to develop systems that try to approach the intellectual and interaction capacities of a human being. Intelligent sensor nodes as things could therfore interact with each other and interchange knowledge in an autonomous way. Concepts related to the artificial intelligence are exploited to create a new methods in computing which involves issues as planning, learning and decision making in a distributive manner. A sensor node as object can be described as a specialized transducer with a dedicated communication infrastructure to monitor

and record parameters from different areas and locations such as temperature, pressure, humidity, illumination intensity, wind speed and direction, sound intensity, vibration intensity, chemical concentration, power line voltage, vital body function and pollutant levels. They can be visioned as extremely basic and small computers regarding their components and interfaces. The typical equipment of a sensor node as, described in the next section, consists mainly of a radio transceiver, a small micro-controller, a memory space and a battery as power supply source.

1.3.1 Smart sensors as IoT devices

Recent advances in the world of microelectronics and wireless technology have led to the development of small and smart sensors with processing capabilities and wireless communications skills. There are many applications that rely on the use of networked sensors to accomplish control and monitoring missions sush as rapid detection of forest fires [1, 2], floods [3], monitoring of river pollution [46, 4] or the surveillance of large buildings such as bridges [5]. This is also the case for understanding natural phenomena such as volcano eruptions [47] and landslides [48].

In fact, smart sensors are an important part of things that are involved in the IoT field. Four main components form the basis of a sensor architecture [49] as in Fig. 1.12.

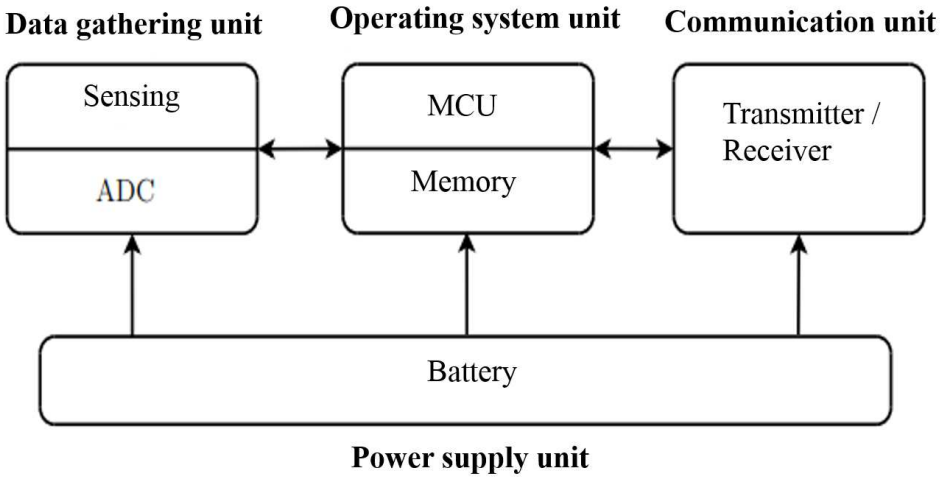

Figure 1.12: Typical sensor node architecture

Theses components are managed by an operating system that uses drivers, protocols and

algorithms. The role of each component are described as follows:

- The operating system which is divided into two units, a controller unit and a memory unit. The controller unit can be based on a microcontroller chipset (MCU) or an ASIC (Application Specific Integrated Circuit). In the first case, great flexibility is guaranteed because the MCU can switch between many operating modes such as off, sleep, idle and active modes. In the sleep mode, the MCU deactivates most of its internal components and goes into the active mode when an external event has occurred (interrupt). The MCU, in idle mode, is also inactive, but many of its internal components are still activated, such as its internal clock and timer. In the active mode, the MCU turn on all of its components. The flexibility offered by the MCU comes at the price [50, 51] of a tradeoff between energy-efficiency and performance. By replacing the MCU with an ASIC, in which the hardware is designed to meet special needs, the loss of flexibility is converted into both high performance and energy efficiency.

- The communication unit is responsible for making the sensor capable of transmitting and receiving data from another device nearby (sensors, sink,...). Wireless communication media are commonly used rather than wired. The communication channels exist in radio frequency, optical or acoustic form [52]. The radio frequency form is the most commonly used technique for wireless sensor communications. The radio transceiver generally has a single channel with low data rate that operates in unlicensed bands of $868 - 870$ MHz in Europe, $902 - 928$ MHz in Americas or near 2.4 GHz as a global band. In wireless sensors, most of the energy is consumed in the radio transceiver. A comparison between the cost of calculation and communication energy has shown that transmitting a bit over a distance of 100 m is equivalent to executing 3000 instructions [53].

- The sensing unit translates physical phenomena into an electric signal. Sensors can be divided into categories based on the physical phenomenon they are intended to measure. Various phenomena can be managed by sensors such as temperature, moisture, light, acoustic and noise, water level, presence and proximity, imagery,

etc. The sensors can also be classified an analog or digital devices according to the outputs produced. An analog sensor must be connected to an analog-digital-conversion device (ADC), which is not the case for a digital sensor.

- The power supply unit supplies all components of the sensor node. The battery is the most used energy source. As an energy reservoir, the available energy decreases as long as it is consumed. Energy harvesting which is used to restore energy into rechargeable batteries, is currently a major research issue in order to adapt it to the needs of sensor nodes [54].

1.3.2 Wireless sensor networks and artificial intelligence

An intelligent sensor is a sensor that changes its internal behavior to optimize its ability to collect environmental data and to communicate with a base station or host system. The widely used artificial intelligence techniques in industries are mainly Machine learning, Fuzzy Logic and Artificial Neural Network. The wireless intelligence of sensors are the result of embedding Intelligent structures in the Wireless Sensor Networks architecture. Resource management is an important part of a middleware solution for WSN, including initial sensor selection and task assignment, as well as real-time adaptation of assigned tasks/resources [55].

Sensor nodes can be described as small and extremely basic computers in terms of their components and their interfaces. Although each of these devices has a very small capacity, their processing capabilities become important when working as an aggregate [56]. Distributed intelligent sensor networks can be viewed as a system composed by multiple agents (sensor nodes), with sensors working together and forming a collective system whose function is to collect data from the physical variables of the systems. Thus, sensor networks can be considered as multi-agent systems or as artificial organized groups that can perceive their environment through sensors [55].

In recent years, there has been a growing interest in the field of artificial intelligence and their methods to solve the constraints of Wireless sensor networks, create new algorithms and new applications for WSN. Harnessing the potential of reinforcement learning fo-

cuses a significant amount of research on artificial intelligence applied to wireless sensor networks, including node-to-node communications. Energy consumption, lifetime of the connectivity network, data coverage and fidelity represent the main research objectives in the field of WSN.

The most important resource in the case of distributed embedded systems is its energy and how to get the most out of it. The main question is how to adapt the tasks to be performed to the resources available throughout the runtime period in order to obtain optimal gains (rewards) and a long network lifetime in a context characterized by limited energy capacity.

Thus, the parameters whose use must be optimized are mainly the service life of the network and the energy consumed for maximum utility for the sensor nodes. In this particular case, machine learning provides optimized strategies for managing the energy inside the batteries and maximizing the number of successful data transmissions, especially with solutions from the fields of reinforcement learning and dynamic programming.

Finally, the key to IoT's success lies in the reliability of the power supply, which is battery-based. In fact, batteries have always been the real and viable solution to supply IoT devices with the energy they need to operate. Given their importance, batteries will be the subject of detailed study in the next chapter.

1.4 Conclusion

This chapter has brought a brief description of the historic evolution of the IoT devices inside a competitive market place. IoT protocol stack layers was presented and described according to the IEEE standards including protocols such as CoAP, UDP, 6LoWPAN, IEEE 802.15.4 and others. This chapter has also introduced the concept of machine learning, as an important branch of artificial intelligence, with its different forms namely the supervised, unsupervised and reinforcement learning approaches, and has discussed matters such as the combination between Artificial Intelligence and IoT, which could bring a significant added value to the quality and the performance of wireless node communications, especially in the context of wireless sensor networks in unknown environments.

Also in this chapter, artificial intelligence for IoT devices has been presented and discussed. Indeed, artificial intelligence is shaping the future of the Internet of Things (IoT). This concept allows the implementation of significant intelligence within the connected objects themselves. The importance of artificial intelligence approaches are discussed in order to enable the emergence of a new generation of intelligent communication networks. Today, smart sensor networks represent a turning point in the history of wireless sensors, especially for remote monitoring of events in areas such as health, the military, forest integrity or the prediction of seismic activity and volcanoes. In addition to their use in terms of home electronics and utilities, smart devices and sensors are also emerging as gadgets for cell phones or tablets.

Chapter

2 Reinforcement learning: A literature review

In artificial intelligence, more precisely in automatic learning, learning by reinforcement consists, for an autonomous agent, in learning the actions to be undertaken, from experiences, in order to optimize a quantitative reward during a sequence. The agent is immersed in an unknown environment, and makes decisions based on his current state. In return, the environment provides the agent with a reward, which can be positive or negative. The agent seeks, through iterative experiments, an optimal decision-making behaviour called policy or strategy, which is a function associating to the current state an action to be executed, in the sense that it maximizes the sum of all the rewards obtained over time.

2.1 Markov decision process

For reinforcement learning, the Markov decision process (MDP) is the most appropriate mathematical tool to formally describe an environment in such a way that it becomes understandable. This environment is supposed to be fully observable in order to be able to read all components of the current system states so that no information is hidden from the learning agent. The current state of the system, mentioned above, is chosen so as to be able to fully describe the process under consideration. What is interesting at this level is that almost all reinforcement learning problems can be formalized as MDPs.

2.1.1 Markov Processes

The main property of the Markov Process (MP) is that what happens later in an environment depends only on its current state. Thus, the future is independent of the past given the present. In Markovian cases, the current state S_t captures all relevant historical information, which means that once the present is known, we can ignore all history, i.e. this state is a sufficient statistic of the future. A state S_t is Markov if and only if the Equ. 2.1 is satisfied.

$$Prob\left[S_{t+1}|S_t\right] = Prob\left[S_{t+1}|S_1, S_2, ..., S_t\right] \tag{2.1}$$

For a Markov state s and successor states s', the state transition probability is defined as in Equ. 2.2.

$$P_{ss'} = Prob\left[S_{t+1} = s'|S_t = s\right] \tag{2.2}$$

State transition matrix defines transition probabilities from all states s to all successor states s' as given in Equ. 2.3.

$$P = \begin{bmatrix} P_{11} & ... & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & ... & P_{nn} \end{bmatrix} \tag{2.3}$$

Where each row of the matrix sums to 1. A Markov process (Markov chain) is a tuple $\langle S, P \rangle$ with the Markov property given as follows:

- S is a finite set of states;
- P is the state transition probability matrix.

2.1.2 Markov Reward Processes

A Markov Reward Process (MRP) is a Markov chain with a value judgment based on the rewards accumulated in a particular sequence of states. In fact, in reinforcement learning, we are primarily concerned with maximizing the immediate rewards. A MRP is a tuple $\langle S, P, R, \gamma \rangle$ with properties as follows:

- S is a finite set of states;
- P is the state transition probability matrix;
- R a reward function, $R_s = Exp\,[R_{t+1}|S_t = s]$;
- γ is a discount factor, $\gamma \in [0, 1]$.

What we're interested in is the total rewards obtained by calculating the return G_t. Where G_t represents the total discounted rewards from time-step t. G_t is the objective obtained as indicated in Equ. 2.4.

$$G_t = R_{t+1} + R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.4}$$

The discount $\gamma \in [0, 1]$ is the present value of future rewards. The value of receiving a reward R after $k + 1$ time-steps is $\gamma^k R$. This value gives the immediate value of delayed rewards so that:

- γ close to 0 means leads to a so called "myopic" evaluation;
- And γ close to 1 leads to a "far-sighted" evaluation.

Discounted rewards are used for certain special needs, as for:

- Avoiding infinite returns;
- Cases of financial rewards, where immediate reward may earn more interest than delayed rewards;
- Human behavior that shows preference to immediate rewards.

But it is possible to use undiscounted Markov Reward Processes (i.e. $\gamma = 1$) , e.g. if all sequences terminates.
The long term value of a state S of a MRP is given by the so called value function (v_s) as defined in Equ. 4.5.

$$v_s = Exp\,[G_t|S_t = s] \tag{2.5}$$

The value function can be divided into two parts, an immediate part representing the immediate obtained reward and a discounted reward representing the delayed part, as shown in Equ. 4.6.

$$\begin{aligned} v_s &= Exp\left[G_t|S_t = s\right] \\ &= Exp\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s\right] \\ &= Exp\left[R_{t+1} + \gamma G_{t+1}|S_t = s\right] \\ &= Exp\left[R_{t+1} + \gamma v_{t+1}|S_t = s\right] \end{aligned} \tag{2.6}$$

This is the Bellman equation which can also be expressed in matrix form as in Equ. 4.7.

$$\boldsymbol{V} = R + \gamma P \boldsymbol{V} \tag{2.7}$$

Where $\boldsymbol{V}$ is a colomn vector with one entry per state as in Equ. 4.8.

$$\begin{bmatrix} v1 \\ \vdots \\ v2 \end{bmatrix} = \begin{bmatrix} R1 \\ \vdots \\ R2 \end{bmatrix} + \gamma \begin{bmatrix} P_{11} & ... & P_{1n} \\ \vdots & \ddots & \vdots \\ P_{n1} & ... & P_{nn} \end{bmatrix} \begin{bmatrix} v1 \\ \vdots \\ v2 \end{bmatrix} \tag{2.8}$$

The Bellman equation is linear and can be solved directly for small MRP's problems with a computational complexity of $0(n^3)$ as in Equ. 4.9.

$$\boldsymbol{V} = (1 - \gamma P)^{-1} R \tag{2.9}$$

2.1.3 Markov Decision Processes

A Markov Decision Process (MDP) is a MRP with decision making power. In addition to the fact that all MDP states are Markovian, the MDP are particularly important for strengthening learning objectives. They are practically used in 90% of cases of reinforcing learning problems. MDP are tuples such as $\langle S, A, P, R, \gamma \rangle$ with properties as follows:

- S is a finite set of states;
- A is a finite set of actions;

- P is the state transition probability matrix, and we note $P^{a}_{ss'} = Prob\left[S_{t+1} = s' | S_t = s, A = a\right]$;
- R a reward function, $R^{a}_{s} = Exp\left[R_{t+1} | S_t = s, A = a\right]$;
- γ is a discount factor, $\gamma \in [0, 1]$.

We define a policy π as a distribution over actions given states, as in Equ. 2.10.

$$\pi(a|s) = Prob\left[A_t = a | S_t = s\right] \tag{2.10}$$

A particular finite MDP is defined by its state and action sets and the one-step dynamics of the environment. Given any state and action, s and a, the probability of each possible next state (transition probabilitie), s', is given in Equ. 2.11.

$$p(s'|s, a) = Prob\left[S_{t+1} = s' | S_t = s, A_t = a\right] \tag{2.11}$$

Similarly, given any current state s and action a, together with any next state, s', the expected value of the next reward r is given in Equ. 2.12.

$$r(s, a, s') = Exp\left[R_{t+1} | S_t = s, A_t = a, S_{t+1} = s'\right] \tag{2.12}$$

These quantities, $p(s'|s, a)$ and $r(s, a, s')$ completely specify the most importance aspects of the dynamics of a finite MDP (only information about the distribution of rewards around the expected value is lost).

The value function of states (denoted v) and action-state pairs (denoted q) estimate the quality of the decision policy for the agent in the case of a reinforcing learning problem and the quality of the performance of a given action in a given state respectively. The reward the agent can expect will then depend on the actions taken. Accordingly, value function of state is defined with respect to particular policies as in Equ. 2.13.

$$v_\pi(s) = Exp_\pi\left[G_t | S_t = s\right] = Exp_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right] \tag{2.13}$$

Where $v_\pi(s)$ is the obtained value function when starting from state s and following policy π thereafter.

And, the value function of an action-state pair q is defined with respect to particular policies as in Equ. 2.14.

$$q_\pi(s,a) = Exp_\pi\left[G_t|S_t = s, A_t = a\right] = Exp_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s, A_t = a\right] \tag{2.14}$$

Where $q_\pi(s,a)$ is the obtained value function when starting from state s, taking action a and then following policy π thereafter. $q_\pi(s,a)$ is called the action-value function for policy π.

For solving large MDP problem cases, many iterative methods can be used so as:

- Dynamic programming methods: like policy evaluation, policy improvement, policy iteration and value iteration;
- Reinforcement learning methods: like Q-learning algorithm, Sarsa, QSarsa, ...

2.2 Model-based control with Dynamic Programming

This section describes a fundamental class of methods for solving finite Markov decision problems in model-based cases using solutions from the field of dynamic programming (DP). In fact, the term DP refers to a set of algorithms that can be used to calculate optimal policies based on a known model of the environment as Markov decision process (MDP). Dynamic programming brings together a set of methods to complex problems, usually by breaking them down into sub-problems. It first solves the sub-problems and then combines the solutions obtained. The term dynamic refers to a temporal or sequential component of the problem and the term programming leads to an optimization process to find an optimal program or policy. Dynamic programming provides an essential basis for understanding methods in the field of reinforcement learning, which is a branch of machine learning. Conventional DP solutions are of little interest for solving real problems because of their assumption of a perfectly known model and the long computation time for large problems. However, all other methods that would be proposed in this work can be seen as attempts to achieve the same effect as with the use of DP, without assuming a perfect known model of the environment and with as little computational effort as possible. From

the beginning of this chapter, the study environment is assumed to be a finite MDP. Dynamic programming is a model-based approach because we assume that the sets of states and actions are known, finite and that their dynamics are given by a known series of transition probabilities as in Equ. 2.11. The rewards generated are given as in Equ. 2.12 for all $s \in S$, $a \in A$ and $s' \in S^+$. Where $S^+ = S +$ is a terminal state ending the episodic state problem.

2.2.1 Policy Evaluation

At this level of analysis, solving the control problem is targeted by using dynamic programming methods. Given an MDP as $\langle S, A, P, R, \gamma \rangle$ and a policy π, the output of the optimization process will be both the optimal state value function v_{π^*} and the optimal control policy π^* to be applied. In general, the main idea of dynamic programming is the use of value functions to structure and organize the search for optimal policies. Once optimal value functions, as v_* and q_{π^*} are obtained, the optimal policy are then easily deduced. v_* and q_* are given in Bellman equations forms as in Equ. 2.15 and Equ. 2.16 respectively.

$$\begin{aligned} v_*(s) &= \max_{a \in A} Exp\left[R_{t+1} + \gamma v_*(s') | S_t = s\right] \\ &= \max_{a \in A} \sum_a \pi(s|a) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v_\pi(s')\right], \end{aligned} \tag{2.15}$$

Or

$$\begin{aligned} q_*(s,a) &= Exp\left[R_{t+1} + \gamma \max_{a' \in A} q_*(s',a') | S_t = s, A_t = a\right] \\ &= \sum_a \pi(s|a) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma \max_{a' \in A} q_*(s',a)\right], \end{aligned} \tag{2.16}$$

For all state $s \in S$, $a \in A$ and $s' \in S^+$.

The objective of this section is to evaluate a given policy π. The evaluation process will be based on an iterative application of the Bellman equations as described above. For all states s and at each iteration $k+1$, the state function value $v_{k+1}(s)$ is updated from $v_k(s')$, where s' is the successor state of s. By the way, the convergence of the value of the state function to v_π is proved by the contraction mapping theorem [57]. Computing the state

value function v_π for any policy π is called *Policy Evaluation* in Dynamic programming field. Based on equations below we obtain v_π as in Equ. 2.17.

$$\begin{aligned} v_\pi(s) &= Exp_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t=s\right] \\ &= Exp_\pi\left[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+...|S_t=s,\right] \\ &= Exp_\pi\left[R_{t+1}+\gamma v_\pi(s')|S_t=s,\right] \\ &= \sum_a \pi(s|a)\sum_{s'} p(s'|s,a)\left[r(s,a,s')+\gamma v_\pi(s')\right], \end{aligned} \tag{2.17}$$

Where $\pi(s|a)$ is the probability of being in state s and taking an action a from policy π. Subscripting the expectation "Exp" by π means that it is conditional on compliance with policy π. Consider a sequence of value state approximations v_0, v_1, v_2,... each mapping S^+ to $\mathbb{R}$. The initial approximation value v_0 will be fixed arbitrarily, but in case of terminal state, if any, it is fixed to 0, and all successive approximation value are obtained by updating the Bellman equation for v_π in Equ. 2.17 as in Equ. 2.18.

$$\begin{aligned} v_{k+1}(s) &= Exp_\pi\left[R_{t+1}+\gamma v_k(s')|S_t=s,\right] \\ &= \sum_a \pi(s|a)\sum_{s'} p(s'|s,a)\left[r(s,a,s')+\gamma v_k(s')\right], \end{aligned} \tag{2.18}$$

It is proven that the sequence $\{v_k\}$ converges to v_π as $k\to\infty$ [58]. This algorithm is called iterative policy evaluation and is given in Algorithm 1 [59].

2.2.2 Policy Improvement

The main reason to evaluate a policy π is to help find better policies. For some state s, the question to answer is whether or not to change the policy and choose action $a\neq\pi(s)$. We know how good to follow the policy π by evaluating v_π, but would be a better or worse to change the policy?. There is one way to find out the answer by selecting an action a in state s and then following the existing policy π. The obtained value for that way of behaving is given in Equ. 2.19.

$$\begin{aligned} q_\pi(s,a) &= Exp_\pi\left[R_{t+1}+\gamma v_\pi(s')|S_t=s,\right] \\ &= \sum_{s'} p(s'|s,a)\left[r(s,a,s')+\gamma v_\pi(s')\right], \end{aligned} \tag{2.19}$$

Algorithm 1 Iterative Policy Evaluation Algorithm

Input π, the policy that will be evaluated

Initialize the array of $v(s)$, for example put $v(s) = 0$ for all states $s \in S$

Repeat

$\Delta \leftarrow 0$

for each state $s \in S$ **do**

$temp \leftarrow v(s)$

$v(s) \leftarrow \sum_a \pi(s|a) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v_k(s')\right]$

$\Delta \leftarrow max(\Delta, |temp - v(s)|)$

end for

Until $\Delta < \theta$ (a small positive number)

Output $v \approx v_\pi$

When $q_\pi(s,a)$ is greater than $v_\pi(s)$, it is preferable to select the action a once at state s and then to follow policy π rather than if the policy π is followed all the time. The *Policy Improvement* theorem states that considering two policies π and π' such as $q_\pi(s, \pi'(s)) \geq v_\pi(s)$ for all $s \in S$, then the policy π' is as good as, or better than, π. That is, policy π' must obtain equal or greater expected reward from all states, which means that $v_{\pi'}(s)$ must be equal or greater than $v_\pi(s)$ (i.e. $v_{\pi'}(s) \geq v_\pi(s)$). So given a policy and its value function, a change of the action at a single state can be easily evaluated. for all states and all possible actions, it is natural to select at each state the action that maximizes the obtained $q_\pi(s,a)$ and thus to consider the greedy policy, π', given in Equ. 2.20.

$$\begin{aligned} \pi'(s) &= \underset{a \in A}{\text{argmax}} q_\pi(s,a) \\ &= \underset{a \in A}{\text{argmax}} Exp\left[R_{t+1} + \gamma v_\pi(s') | S_t = s, A_t = a\right] \\ &= \underset{a \in A}{\text{argmax}} \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v_\pi(s')\right] \end{aligned} \tag{2.20}$$

2.2.3 Policy Iteration

The π policy which was improved using v_π, is now replaced by a better policy, π'. $v_{\pi'}$ can then be computed to improve π' and an even better π'' can then be obtained and so on as shown in Fif. 2.1. Where "E" and "I" stand for policy evaluation and policy

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

Figure 2.1: Sequences of policy evaluations and improvements

improvement respectively. Each policy brings improvement from its previous one unless it is optimal. Since the considered problem is a finite MDP, this process must converge to an optimal policy π^* and a value function v_* in a finite number of iterations.

Policy iteration, as it is called in the literature, is the process of finding the optimal policy. The algorithm 2 [59] describes in details different phases of the Policy Iteration algorithm.

2.2.4 Value Iteration

Each iteration of the Policy Improvement algorithm uses policy evaluation in such a way that an extended iterative calculation requiring multiple scans across all states is performed. In fact, it may be possible to truncate policy evaluation sequences without losing the guarantee of convergence in the policy improvement process. This algorithm for calculating optimal policies is called *Value Iteration* in the literature. The Value Iteration algorithm can be written in a particular way so that truncated policy evaluation and policy improvement stages can be combined. Algorithm 3 [59] gives the whole process of the optimal policy calculation based on Value Iteration. Faster convergence can then be obtained in this case compared to what one could have obtained in the case of policy improvement.

Algorithm 2 *Policy Iteration Algorithm*

1.Initialization

Choose $v(s) \in \mathbb{R}$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

Initialize the array of $v(s)$, for example put $v(s) = 0$ for all states $s \in S$

2.Policy Evaluation

Repeat

$\Delta \leftarrow 0$

for each state $s \in S$ **do**

$temp \leftarrow v(s)$

$v(s) \leftarrow \sum_a \pi(s|a) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v_k(s')\right]$

$\Delta \leftarrow \max(\Delta, |temp - v(s)|)$

end for

Until $\Delta < \theta$ (a small positive number)

3.Policy Improvement

$policy - stable \leftarrow true$

for each state $s \in S$ **do**

$temp \leftarrow \pi(s)$

$\pi(s) \leftarrow \underset{a \in A}{\operatorname{argmax}} \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v(s')\right]$

If $temp \neq \pi(s)$, then $policy - stable \leftarrow$ false

end for

If $policy - stable$, then stop and return v and π; else go to 2

Algorithm 3 Value iteration Algorithm

Initialize the array of $v(s)$, for example put $v(s) = 0$ for all states $s \in S$

Repeat

$\Delta \leftarrow 0$

for each state $s \in S$ **do**

$temp \leftarrow v(s)$

$v(s) \leftarrow \sum_a \pi(s|a) \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v_k(s')\right]$

$\Delta \leftarrow \max(\Delta, |temp - v(s)|)$

end for

Until $\Delta < \theta$ (a small positive number)

Output the optimal policy, π^*, such that:

$\pi^*(s) = \underset{a \in A}{\operatorname{argmax}} \sum_{s'} p(s'|s,a) \left[r(s,a,s') + \gamma v(s')\right]$

2.3 Reinforcement learning

Compared to the dynamic programming methods presented in the previous sections, in which the agent knows in advance the transition function and the reward function of the MDP's problem. Reinforcement learning methods allow to deal with the situations in which the functions of transition and reward are not known in advance. This chapter is devoted to the treatment of this complex problem in unknown environments. The mathematical formalism of reinforcement learning as it will be presented in this chapter was first developed in 1988 by Sutton [60] and later by Watkins [61], who linked their works to the framework of optimal control theory, as proposed by Bellman in 1957, and to the notion of Markov decision processes (MDP) [62]. There are two main important learning models in reinforcement learning:

- Q-learning Algorithm;
- Sarsa Algorithm;
- A mix of both methods can also be used which is called Q-Sarsa Algorithm.

2.3.1 Q learning algorithm

One of the most important reinforcement learning breakthroughs was the development of the control algorithm known as Q-learning. It is a form of model-free reinforcement learning and a simple way for agent to learn how to act in controlled Markovian domains by successively improving the evaluations of particular actions at particular states. Q-learning proceeds as in Temporal-Difference learning methods developed by Sutton in [63, 64] where the agent tries an action in particular state, and evaluates the reward or penalty it receives as consequences as well as the value estimation of the state to which it is taken. Q-learning converges to the optimal action-state function value q_* with probability 1 as long as all actions are repeatedly sampled and applied to all states [65].

The form of one basic Q-learning step is defined as in Equ.2.21.

$$\mathrm{q}(S_t, A_t) \leftarrow \mathrm{q}(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \times \max_{a \in A}(\mathrm{q}(S_{t+1}, A_t)) - \mathrm{q}(S_t, A_t) \right] \tag{2.21}$$

In this case, the optimal action state value function q_* is directly estimated by the learned action-value function Q whatever the policy being followed is.

The procedural form of Q-learning is detailed in Algorithm 4.

Algorithm 4 Q-learning Algorithm

Initialize $\mathrm{q}(s, a)$, for example put $\mathrm{q}(s, a) = 0$ for all states $s \in S$ and actions $a \in A$

Repeat (for each episode)

Initialize s

Repeat (for each step of episode)

Choose a for s using policy derived from Q (e.g. $\epsilon - greedy$)

Take action a, observe r and s'

$\mathrm{q}(s, a) \leftarrow \mathrm{q}(s, a) + \alpha \left[r + \gamma \max_{a \in A} \mathrm{q}(s', a) - \mathrm{q}(s, a) \right]$

Put $s \leftarrow s'$

Until s is a terminal state

2.3.2 Sarsa algorithm

The Sarsa algorithm takes its name after the tuple $< s, a, r, s', a' >$ which represents the succession of phases that update the value of the state and the action as shown in Fig. 2.2.

Sarsa algorithm is quite similar to Q-learning with few differences. The main difference between them is that Sarsa is an on-policy algorithm whereas Q-learning is an off-policy algorithm. This implies that Sarsa updates the action state value Q by choosing actions based on the current policy of the agent, whereas Q-learning takes its actions from the $epsilon - greedy$ policy which is different from the policy being followed. In Sarsa, for the next state S_{t+1}, the action A_{t+1} that is performed comes directly from the current policy being followed. The form of one basic Sarsa learning step is defined as in Equ.2.22.

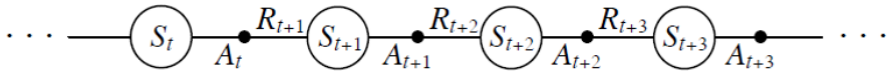

Figure 2.2: Sarsa named after the sequence State-action-reward-state-action

$$\mathrm{q}(S_t, A_t) \leftarrow \mathrm{q}(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \times \mathrm{q}(S_{t+1}, A_{t+1}) - \mathrm{q}(S_t, A_t)\right] \tag{2.22}$$

The Sarsa algorithm is shown in procedural form in Algorithm 5.

Algorithm 5 Sarsa Algorithm

Initialize $\mathrm{q}(s, a)$, for example put $\mathrm{q}(s, a) = 0$ for all states $s \in S$ and actions $a \in A$

Repeat (for each episode)

Initialize s

Choose a for s using policy derived from Q (e.g. $\epsilon - greedy$)

Repeat (for each step of episode)

Take action a, observe r and s'

Choose a' for s' using policy derived from Q (e.g. $\epsilon - greedy$)

$\mathrm{q}(s, a) \leftarrow \mathrm{q}(s, a) + \alpha \left[r + \gamma \mathrm{q}(s', a') - \mathrm{q}(s, a)\right]$

Put $s \leftarrow s'$ and $a \leftarrow a'$

Until s is a terminal state

2.3.3 Q-learning versus Sarsa algorithm

In fact, when Q-learning, as an off-policy, does not follow the policy it is learning, Sarsa follows exactly the policy being learned. Actually, as expecting in the Q updating equa-

tions in 2.21 and 2.22 , the cost of exploration (according to the used $epsilon - greedy$ policy) is only considered in Sarsa algorithm as Q-learning doesn't take it into account. The question about whose the better between the two algorithms is quite hard to answer but they both have advantages and disadvantages. Based on the cliff walking problem as it was introduced by Sutton and Barto [59], some of these advantages and disadvantages can be found out. The cliff walking problem as it is illustrated in Fig. 2.3, consists of finding the optimal route for getting from the start point to the goal. In the case of Fig.2.3, the optimal route will that the agent makes 10 steps towards the goal position. Nevertheless, if the agent fall of the cliff it will receive a high negative reward. Even if the agent has found the optimal action state value Q^*, it will sometimes fall off the cliff because it uses an $\epsilon - greedy$ policy.

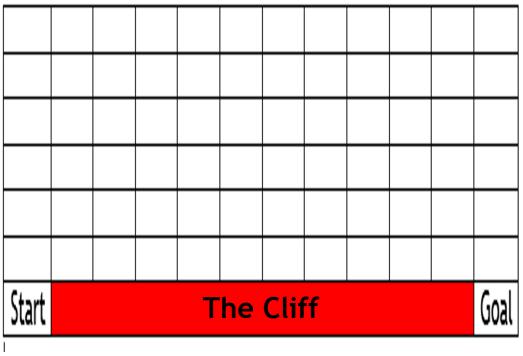

Figure 2.3: Sarsa named after the sequence State-action-reward-state-action

What Sutton and Barto observed with the 10-step cliff is that the algorithmic agent Sarsa never falls off the cliff but chooses a route to reach the goal position that was farther away from the cliff. This means that the algorithmic agent Sarsa both learns how to avoid the cliff and the road to the goal. The Q-learning agent can find the optimal path faster but with the cost of falling off the cliff several times. Moreover, it seems that Q-learning agent would find the optimal policy for the illustrated cliff walking problem (i.e. the 10-step problem), unlike agent Sarsa who will not, which is not entirely untrue. However, for larger problems, such as a problem walking on a cliff with 10,000 steps, the Q-learning agent will never learn that it is dangerous to approach the cliff and will therefore fall constantly, which means that his learning process will evolve very slowly. On the contrary, Sarsa

agent who can avoid falling off the cliff will therefore quickly learn a good policy to reach the goal position. This may lead someone to say that Sarsa learning agents are better suited for optimizing large problems, to which Sarsa therefore provides quick solutions, while Q learning agents learn faster for smaller problems.

By way of illustration, Kamenetsky [66], explores both Q-learning and Sarsa for the game of othello, which is a complex problem. In fact, for the othello game problem, Sarsa learns faster than Q-learning but its learning curve is unstable. However, the final results were similar for both algorithmic agents. Actually, the instability of the Sarsa curve comes from the integration of exploratory actions in the calculation of the state-action function, which is not the case for Q-learning.

And finally, we stress how important it is to consider canceling the ϵ effect towards the end of the training process. This is particularly valuable in the case of Sarsa-learning, where any bad exploratory action incorporated in the action-state function leads to fluctuating values of Q.

2.3.4 Q-Sarsa algorithm

Even though Q-learning and Sarsa techniques seem to be similar, they still show different advantages and disadvantages as stated earlier. Combining the two techniques would lead to an apparent improvement by forming a Q-Sarsa algorithm [67]. The combination of the two algorithm could use a simple parameter, let say σ (where $\sigma \in [0, 1]$). The update rules in Equ.2.21 and Equ.2.22 can be combined to get the Q-sarsa update rule. If $\sigma = 0$, the update rule will be Q-learning and if $\sigma = 1$ the rule will be Sarsa as given in Equ. 2.23.

$$\mathrm{q}(S_t, A_t) \leftarrow \mathrm{q}(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \left((1-\sigma) \max_{a \in A} \mathrm{q}(S_{t+1}, A_t) + \sigma \mathrm{q}(S_{t+1}, A_{t+1}) \right) - \mathrm{q}(S_t, A_t) \right] \tag{2.23}$$

The Q-Sarsa algorithm is shown in procedural form in Algorithm 6.

Algorithm 6 Q-Sarsa Algorithm

Initialize $\mathrm{q}(s,a)$, for example put $\mathrm{q}(s,a)=0$ for all states $s \in S$ and actions $a \in A$

Repeat (for each episode)

Initialize s

Choose a for s using policy derived from Q (e.g. $\epsilon - greedy$)

Repeat (for each step of episode)

Take action a, observe r and s'

Choose a' for s' using policy derived from Q (e.g. $\epsilon - greedy$)

$$\mathrm{q}(s,a) \leftarrow \mathrm{q}(s,a) + \alpha \left[r + \gamma \left((1-\sigma) \max_{a \in A} \mathrm{q}(s',a) + \sigma \mathrm{q}(s',a') \right) - \mathrm{q}(s,a) \right]$$

Put $s \leftarrow s'$

Put $a \leftarrow a'$

Until s is a terminal state

2.3.5 Exploitation versus exploration in reinforcement learning

A fundamental issue in reinforcement learning algorithms is the balance between exploring new actions and exploiting the information already obtained by the agent. Reinforcement learning, as a model-free optimization method, requires a kind of action-selection strategy that balances exploration and exploitation. A simple definition of exploitation under the action selection approach is to select the action with the highest expected reward on a consistent basis. Exploration is defined as the approach of selecting the best action most often and choosing a random action the rest of the time. In literature, there are several selection strategies that can be used to achieve a good balance between exploration and exploitation. This section reviews the most popular exploration strategies [68] used in reinforcement-based learning such as Boltzmann-Gibbs selection, Frequency-based exploration, Recency-based exploration, Error-based exploration and $\epsilon - greedy$ selection.

Boltzmann-Gibbs Selection

In order to calculate a probability of selecting the actions, Boltzmann-Gibbs distribution uses q(s,a) for each individual action. This probability is given as in Equ. 2.24.

$$Prob(s|a) = \frac{Exp^{q(s,a)/\tau}}{\sum_{a' \in A(s)} Exp^{q(s,a')/\tau}} \tag{2.24}$$

Where $Prob(s|a)$ is the probability for selecting action a in state s and τ is a temperature variable, $\tau > 0$.

Frequency-Based Exploration

The frequency-based exploration strategy uniformly explores all state-action pairs by calculating a value-of-experience function $R_{Exp}(s, a)$ as in Equ. 2.25.

$$R_{Exp}(s, a) = -\frac{Count(s, a)}{K} \tag{2.25}$$

where $Count(s, a)$ is a local counter, counting the times that the action a has been selected in state s, and K is a scaling constant. This exploration strategy will always try to explore the state-action pair that has been visited least frequently. The experience value is always negative, which means that a state-action pair that has never been visited will always have a higher $R_{Exp}(s, a)$ than all the previously visited state-action pairs.

Recency-Based Exploration

The Recency-based exploration strategy will try to explore the state-action pairs that has not been visited for a long time by calculating the experience value function $R_{Exp}(s, a)$ as given in Equ. 2.26.

$$R_{Exp}(s, a) = -\frac{Time(s, a)}{K} \tag{2.26}$$

Where Time(s, a) represents the last time-step where the state-action pair (s, a) was visited. Both frequency- and recency-based exploration can be seen throughout the literature with different functions for the same goal. When Recency-based exploration was introduced by Sutton in [69] he used a value based on the square root since the last

visit to the state-action pair. The same approach is reproduced in a more advanced form by Pchelkin in [70], which incorporates the total number of states. Wiering in [71] and Kolle in [72] define a new recency-based exploration value function, which is based on the current time-step T_{step} instead of $Time(s, a)$.

Error-Based Exploration

The error-based exploration strategy attempts to explore areas of state-action pairs space where the $Q(s, a)$ estimate rises sharply, as this could be an area where the agent could gain positive knowledge about how to improve policy. For this strategy, the experience value function $R_{Exp}(s, a)$ is calculated as given in Equ. 2.27.

$$R_{Exp}(s, a) = q_{n+1}(s_n, a_n) - q_n(s_n, a_n) - K. \tag{2.27}$$

The $\epsilon - greedy$ Selection

The $\epsilon - greedy$ strategy acts by selecting the greedy action $1 - \epsilon$ of the time, and selecting a completely random action the rest of the time, where $0 \leq \epsilon \leq 1$. By the term greedy, we mean an action that has been shown to give the maximum reward based on the preceding values of the action-state value function Q. This approach of action selection is widely used, very easy to implement and in fact works quite well most of the time. This is why we will adopt the $\epsilon - greedy$ selection because of its simplicity and its frequent use in several areas of machine learning.

2.4 Conclusion

In this chapter, we introduced the Markov decision process concept, which is the basis for using dynamic programming and machine learning tools to solve optimization problems. In addition, we have discussed the different approaches that will be considered in the next chapters from the areas of dynamic programming and reinforcement learning.

In a model-based context, it is assumed that sufficient knowledge was available to identify the elements of the MDP model, such as the set of states S and actions A, the probability transition matrix P, the reward function R and the discount factor γ. These elements will be combined to calculate the optimal policy to be followed by using solutions from the field of dynamic programming such as value iteration algorithm. In this case, the optimal policy is calculated at a cost of complexity that depends on the size of each component of the CDM system. The computation time required for the convergence of the algorithms also depends on the size of the system.

In the free model case studies, it is assumed that there is no prior knowledge of the environmental characteristics of the system. In this situation, we start by initiating a learning phase of the embedded system environment in order to acquire a preliminary knowledge of it. To do so, we use techniques from the field of machine learning, precisely from the reinforcement learning toolbox. An intermediate scenario will be introduced in Chapter 5, where we assume that we have a partial knowledge of the system under study which we can exploit to reduce the complexity and duration of calculations before obtaining optimal policies.

Finally, the deduction of optimal policies in the case of a priori known embedded system models is directly obtained from the convergence of the dynamic programming algorithms. For unknown models, the policy design is conducted at the end of the learning process by extracting the greedy actions for high rewards from the obtained action-state value functions. But before moving on to the calculation of optimal policies, it is essential, as detailed in the next chapter, to understand how batteries work and to define a battery model for further use that is best suited to our needs and to the use of dynamic programming and reinforcement learning algorithms.

Chapter

3

Literature review on batteries for embedded systems

The term "battery" was first used to describe an assembly of cannons in artillery units, and then it was used around 1750 to describe, for the first time, a Leyden (or Lieden) jar capacitors which was an antic electrical component that stores a high voltage electrical charge from an external source. Finally, it is in 1801 that this term was applied to an assembly of galvanic cells to obtain a higher power. It is also common to use the word battery even for a single cell [73].

In this chapter, we present the main battery characteristics that must be taken into account in order to obtain accurate battery models. In addition, we will describe some of the battery models cited in the literature, such as the electrical circuit, the analytical, the black box and the stochastic battery models. These models provide a satisfactory description of the batteries, but not all of them are practical for dealing with an energy optimization problem. However, the stochastic model that will be described in section 3.2.5 seems, in our opinion, to be the most suitable for this purpose.

3.1 Batteries for embedded systems

There are many different types of energy systems used by embedded systems. For example, we can identify the capacitors, fuel cells, energy harvesters, batteries and many other sources.

3.1.1 Embedded energy systems

The different types of embedded energy system are summarized and listed in Fig. 3.1. For their convenience, these power solutions can all be used to power IoT devices such as sensor nodes in wireless sensor networks context.

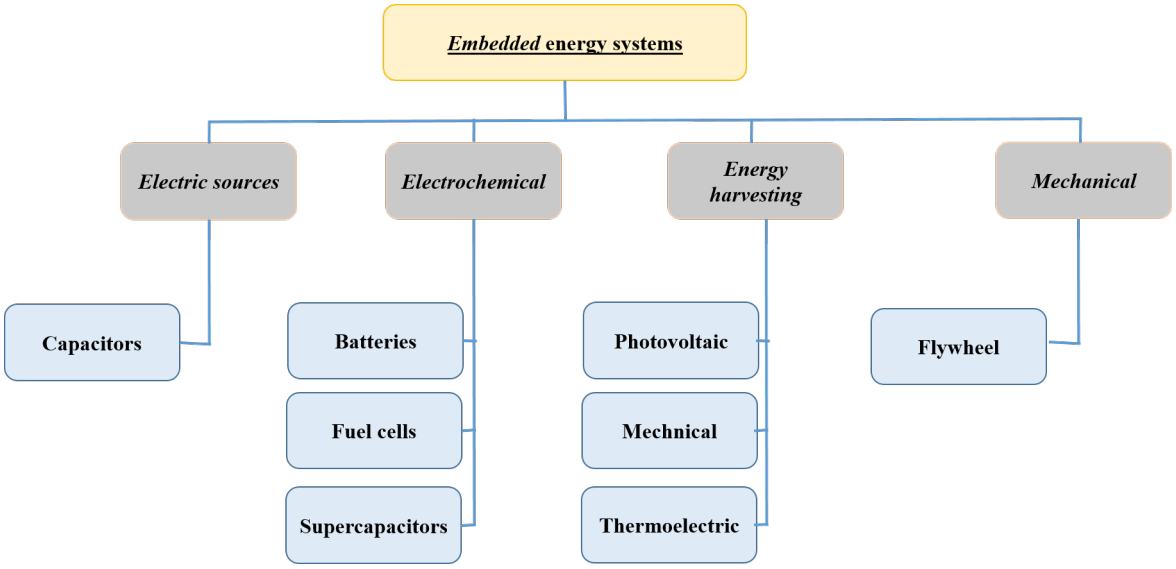

Figure 3.1: Various types of embedded energy systems

According to the Ragon diagram of energy storage devices in Fig. 3.2 [74], even though super-capacitors have a much higher peak power density than batteries, their energy density is still very low. This is why batteries, which have a much higher energy storage capacity, are best suited for low-energy sensor nodes that need to operate for a long period of time.

In fact, embedded devices often depend on batteries for operation and communication. Figure 3.3 gives a view of such embedded devices, where the batteries can be of primary or secondary type. The radio communication part consists of a digital central processing unit (CPU), a radio frequency (RF) unit and a power amplifier (PA), each of these units requiring power supplies of different ranges which are supplied by the power supply unit (PS) [75].

In general, the CPU and receiver consumptions are constant. However, it has been proven that the power amplifier, together with the transmitter, consumes the greatest amount of energy during each transmission. When primary batteries are used, the only way to

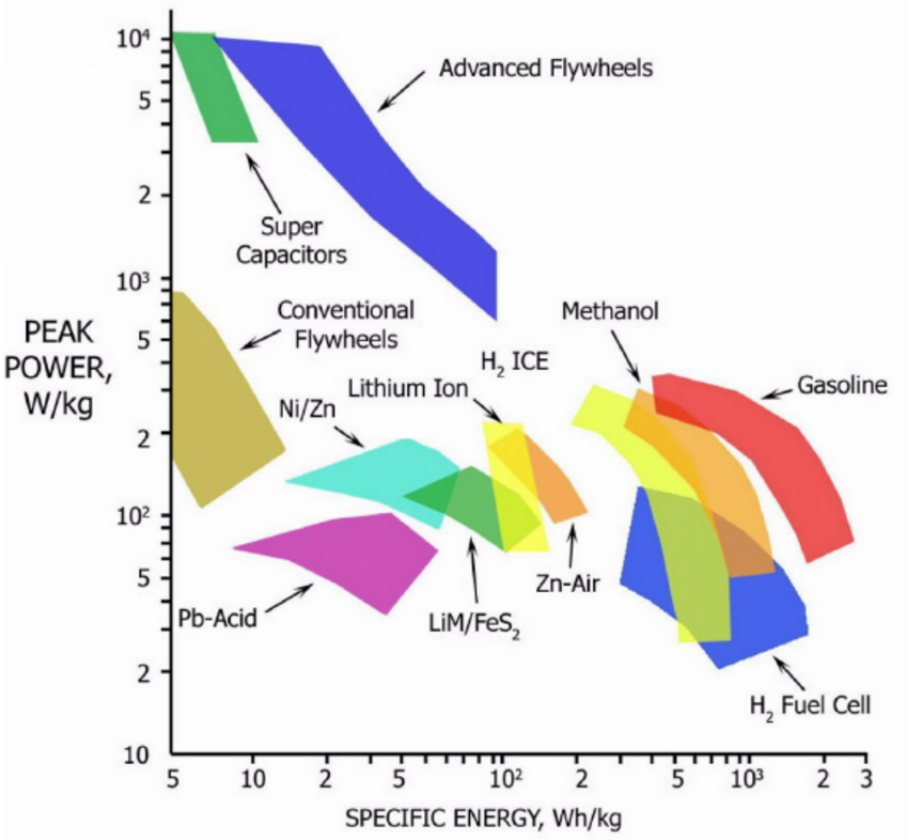

Figure 3.2: Ragone plot of energy storage devices

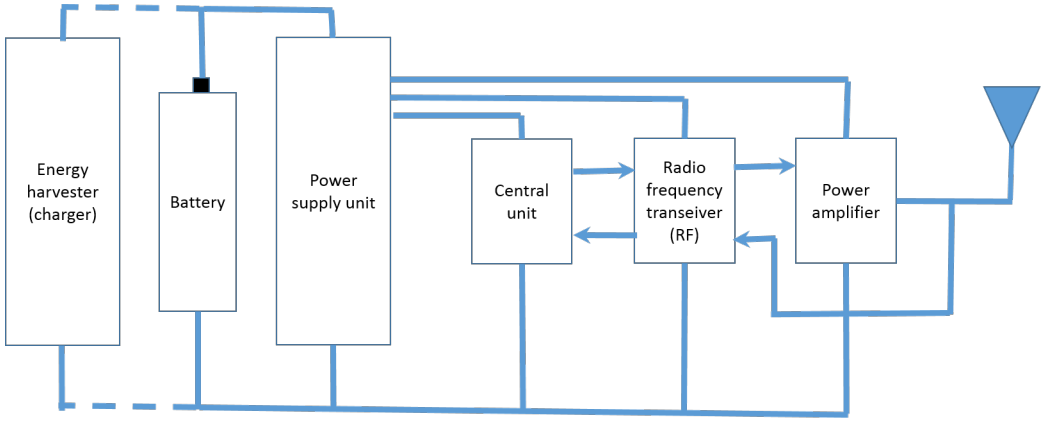

Figure 3.3: The built-in units of an embedded system communication

ensure a continuous supply of power when the batteries are depleted is to replace them. Alternatively, for secondary batteries, when in use, they can rely on an energy harvesting charger to renew their capabilities. It is obvious that the characteristics of batteries, such as their small size, light weight and long life, are the most sought-after and that optimal

management of their energy stock is a crucial issue for embedded system communications. An example of batteries in use for the power supply of the IoT devices is shown in Fig. 3.4.

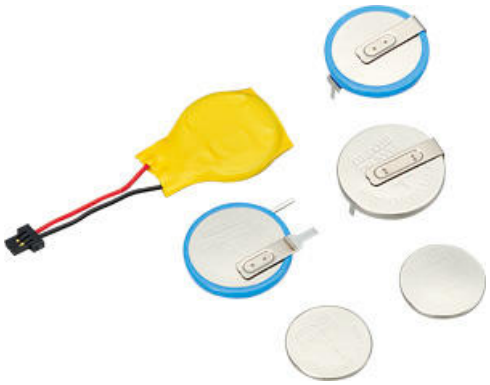

Figure 3.4: A set of battery coin cells in use for IoT

3.1.2 primary batteries

Primary batteries are cells designed to be used once and discarded, and cannot be electrically recharged and reused [76]. Since, the typical power sources for sensor nodes are primary batteries, there is no needs for intermediate voltage conditioning components as their energy are provided at the required voltage levels. The initial energy stored in the battery, since it is the only source of energy for the sensor node, determines the lifetime of the node. Existing Alkaline batteries have commonly an energy density of 1200 J/Cm^3, lithium batteries provide an energy density of 2880 J/Cm^3 and the Zinc-air batteries which have the highest level with an energy density around 3780 J/Cm^3 [77].

3.1.3 secondary batteries

A rechargeable battery or secondary cell is a type of electrical battery which can be charged, discharged into a load, and recharged many times, as opposed to a primary battery, which is supplied fully charged and discarded after use [78].

Since the stored energy in the primary batteries are finite, battery replacement becomes essential for long term applications. Unfortunately, in most application cases the battery

replacement is not feasible. Therefore, the use of energy harvesters to collect packets of energy from ambient sources can be a practical solution to mitigate this problem. Since stable energy cannot be supplied directly from ambient sources, secondary batteries are then used to store the collected energy for later use.

As with primary batteries, it is the internal chemistry that determines the characteristics of secondary batteries. The conventional secondary batteries such as nickel metal hydride (NiMH), nickel-cadmium (NiCd) and nickel-zinc (NiZn) provide high energy densities and good discharge rates but have short cycle lives and adverse memory effects [79]. The NiMH secondary batteries have an energy density of 860 J/Cm^3 and the NiCd secondary batteries provide an energy density of 650 J/Cm^3. It is the Lithium-ion secondary batteries that offer the highest energy density of 1080 J/Cm^3 [77].

3.1.4 Battery behavior description and analysis

As a simple model, a battery acts as a reservoir of chemical energy which is then converted into electrical energy [80]. For batteries, the potential described in volts tends to decrease as the battery capacity is depleted. Battery capacity is measured in milli-ampere-hour (mAh) or ampere-hour (Ah). For example, a capacity of 3 Ah means that the battery can provide 1 A for 3 $hours$ or 3 A for 1 $hour$. Often the battery is not completely empty, but its potential energy is not sufficient to extract the remaining charge, so the battery is said to be empty when its potential or voltage drops below a certain level [76]. For example, in case of a 1.5 V AA-sized cell, the battery is said to be empty when the voltage drops below 0.9 V.

Voltage and capacity are the most important characteristics of a battery. The energy stored in a battery can be obtained simply by multiplying these two elements $V \times C$. Ideally, the capacity remains constant regardless of the battery charge and the voltage remains the same for the battery during its operating time until the voltage drops to zero, which means that the battery is completely discharged. In reality, the capacity effect of the battery makes things happen differently, as the voltage drops during battery discharge and the capacity decreases with higher battery charges.

Two important parameters are mainly used to describe batteries. They are [81]:

- The theoretical capacity T_B which is the total number of charge units located in the electrolyte and electrodes before first use ;
- And the nominal capacity B_{max} is equal to the effective extracted charges from the battery under a specific discharge current till the battery voltage reach the cut-off value where the battery is declared to be fully exhausted and then becomes out of use.

Parameters T_B and B_{max} are both sensitive to the battery materials and the current discharge conditions.

3.1.5 Relation between Discharge current and the nominal capacity

It is the electrochemical reactions at the electrode-electrolyte interface of the battery that lead to the creation of the resulting current [80]. When no charge is connected to the battery, the current is zero and the active charge units are uniformly distributed over the area of the electrode-electrolyte interface [82]. When a charge is connected, the current becomes greater than zero and the charge units at the interface are consumed and replaced by new units from the electrolyte solution through diffusion effect. As the current value grows, the average concentration of active charge units starts to deviate between the electrode and the electrolyte by polarization effect. Consequently, the state of charges at the electrode as well as the operating voltage decrease. Above a threshold current value, the diffusion phenomena become unable to replace the used charge units leading to a depletion of charges at the interface that quickly drops to zero at the cathode. At this point, the operating voltage drops below the cut-off value (V_{cut}) and even if the theoretical capacity has not yet been fully consumed (i.e. an amount of charge units still available in the electrolyte solution), the battery is considered to be discharged.
Consequently, a high level of consumption of the theoretical capacity (energy extracted from the battery) can only be obtained for low current discharge rates. For illustration, in case of a lead acid battery with a nominal capacity of 5 Ah, The extracted capacity is improved by 50% if the discharge current is of $250mA$ instead of $5A$ [83]. For Lithium

batteries, the delivered capacity is increased by 60% when the discharge current is of $125mA$ instead of $1A$ [82].

To illustrate this relationship between discharge current and nominal capacity, Fig. 3.5 shows the behavior of a Lithium Manganese Dioxide Battery (coin cell see Fig. 3.6) with $V_{OC} = 3$ V and $V_{cut} = 2$ V for different values of power drain.

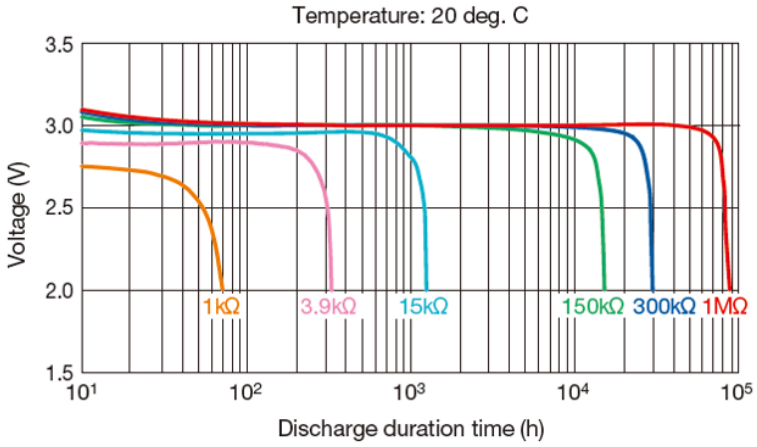

Figure 3.5: Discharge characteristics for coin Type Lithium Manganese Dioxide Battery 20mm x 3.2mm ($CR2032H$) [84]

Figure 3.6: Lithium Manganese Dioxide Battery coin cell CR2032H

3.2 Battery modeling approaches

Battery models have become an essential tool for the design of battery powered systems. In literature, there are mainly five (5) types of models to represent the temporal behavior

of the battery which are the electrochemical models, the black box models, the electrical circuit models, the analytical models and the the stochastic models.

3.2.1 Electrochemical model

The chemical processes that occur in the battery are the basis of the electrochemical models. In fact, these models give a detailed description of the battery's processes. Indeed, as this description gives many details, these models are considered to be the most accurate. However, such a precise description makes the models difficult and complex to configure. For example, the electrochemical models for lithium-ion and lithium batteries that were developed by Fuller, Doyle and Newman [85, 86, 87] are represented by a set of six coupled nonlinear differential equations, and the battery model proposed in [88] is represented as a mixed system of 14 nonlinear partial differential algebraic equations (PDAE) with 14 unknowns, all of which must be solved to find the model to be used for the simulation. Although the electrochemical models are able to provide accurate estimations of the battery behavior [89, 90], their degree of complexity makes them difficult to use for simulation purposes.

In addition, these models require prior knowledge of more than 50 battery-related parameters, such as electrode thickness, initial salt concentration in the electrolyte and overall heat capacity, which are very difficult to determine. While estimates of some parameters can be obtained from the literature, others require extensive experimental research which is not always feasible.

Although the accuracy of electrochemical models can be very high, they are not suitable for dealing with real-time application control strategies. When available, these models are often used for comparison with other models, rather than using experimental results to verify accuracy.

3.2.2 Black box type model

Black box type models use learning methods [91, 92], such as Artificial Neural Networks and Fuzzy Logic, to model batteries and predict their behaviors. Unlike electrochemical

models, black box models do not need to have a complete knowledge of electrochemical phenomena. In fact, on the basis of the data previously collected on the actual operating mode of the batteries, a learning phase must be carried out on this collected data, which are of different types such as current, voltage, temperature, etc. The result of this phase is the obtaining of empirical models produced in the form of black boxes.

These models are able to predict the behavior of the battery based on their own experiences during the learning phases. However, the accuracy of these models is highly dependent on training methods and data [93]. In addition, the uses of black box models are mainly limited to the estimation of the state of charge (SoC) and state of health (SoH) of batteries [94].

Only a few articles have been produced in the literature to predict battery behavior using black box models [91]. These models would not be retained for the continuation of this work because they are not mature enough and are not the best suited to build accurate models of battery behavior.

3.2.3 Electrical-circuit model

Hageman [95] proposed the first electrical-circuit models in 1993. In fact, to simulate alkaline, lead-acid and nickel-cadmium batteries, Hageman used simple PSpice circuits. To do this, the model elements were assumed to be the same for the different types of batteries. The elements of the battery taken into consideration are mainly the capacitor which represents the capacity of the battery, the discharge rate which determines the loss of capacity at a high discharge current, a circuit for discharging the capacity of the battery, a table of correspondence between the voltage and the SoC (State-of-Charge) and a resistor which represents the resistance of the battery. The basic circuitry that is used to model a battery cell are shown in Fig. 3.7. Even if these models are less complex than electrochemical models and therefore have a lower calculation cost, their configuration still requires a great deal of effort, especially the look-up table, which requires a lot of experimental data on the behavior of the battery. In addition, these models are not accurate in predicting and estimating battery life, as they make errors of up to approximately 12% [94].

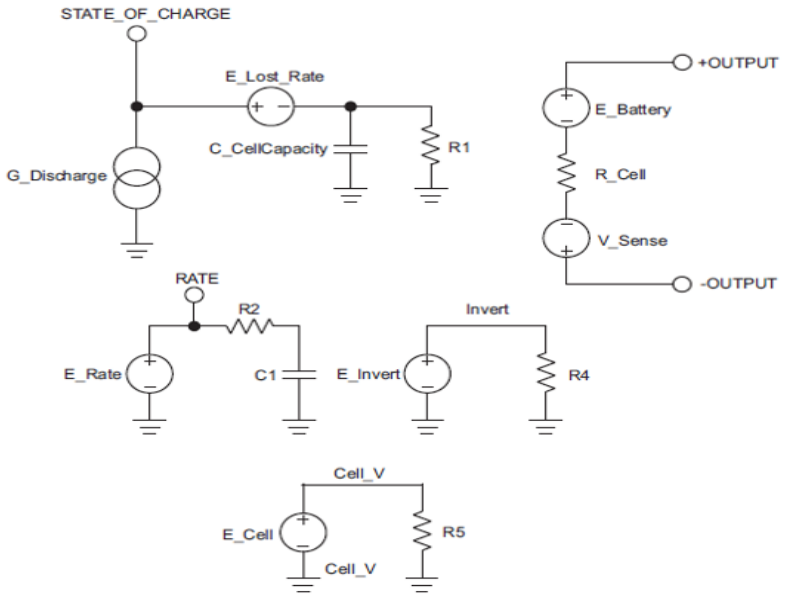

Figure 3.7: Basic functional schematic battery cell model [95]

3.2.4 Analytical model

Battery models are now based on a few equations, which makes analytical models much easier to use than electrical and electrochemical circuit models. Three main analytical models are used in literature which are the Peuker's law model [96], Rakhmatov and Vrudhula model [96] and the Kinetic Battery Model (KiBaM) [97, 98, 99]. However, Peuker's law and the models of Rakhmatov and Vrudhula do not take into account all the phenomena of the battery, in particular the recovery effect and the equations of the kinetic model of the battery do not apply to modern batteries used in mobile devices [94]. As a result, this model is also not adapted to the optimization problem we are dealing with.

3.2.5 Stochastic model

Between 1999 and 2001, Chiasserini and Rao proposed a series of stochastic models that describe the battery based on discrete time Markov chains [81, 100, 101, 102]. Since [81], the authors have proposed two battery models for mobile communication devices. In both these models, a discrete time Markov chain with $B_{max}+1$ states is introduced to describe the battery as given in Fig. 3.8 ($B_{max} = N$) , where the states are numbered from 0 to

N. These numbers refer to the charge units available in the battery. One charge unit is assumed to be the amount of energy required for a basic packet transmission. The number N refers to the amount of charge units directly available for the current data packet transmission.

In [81], the first model given in Fig. 3.8, consider that a charge unit is consumed with probability $a_1 = q$ or recovered with probability $a_0 = 1 - q$ at each time step. When the state 0, also called trapping state, is reached or a maximum of T_B charges are consumed, the battery becomes empty. T_B represents the total number of charge units, available into the electrolyte, and equivalent to the theoretical capacity of the battery where $T_B > N$ (N_S represents a dummy state to show the start of discharge). Furthermore, in Fig. 3.8 the authors assume that, during each time slot, the system transmits only one data packet if available, otherwise it recovers one charge unit. That is why this model is called binary pulsed discharge model. In case of data packet transmission, the energy required by the transmitter power amplifier is obtained from the battery by draining a constant current during the entire duration of the time slot. For this first model, it is possible to determine analytical expressions for all the properties of interest for the embedded system. Properties that can be investigated are like the expected total amount of transmitted packets $\bar{m}_p$ and the obtained gain G from the intermittent discharge compared to a constant discharge profile that are clearly defined by Equ. 3.1. It is proven that an intermittent discharge profile will lead to a gain G that exceeds 1 due to the recovery effect that occurs during idle periods. G can be at most equal to T_B/N. Intermittent discharge profile that outperforms the constants discharge profile since it can exceeds 1 as it approaches T_B/N.

$$G = \bar{m}_p/N \tag{3.1}$$

In the second model, that is proposed for more generalized situations, the authors assume that the system may either transmits one or more data packets or recovers charge unit during each time slot. The amplitude of the impulse is then assumed to be equal to the current required by the transmitter power amplifier times the number of the packets to be transmitted. The impulse may occupy only a fraction of the time slot and the remaining

time of the slot duration should allow the system to recover of one charge unit. This model is called generalized pulsed discharge model and is given in Fig. 3.9. With probability q_i, i charge units are requested in one time slot for performing the transmission of the available packets. During the idle periods, the battery either recovers one charge unit with probability $p_j(f)$, or stays in the same state with probability $r_j(f)$, where j is the state number and f the total number of charges having already been drained from the electrolyte. The recovery probability in state j after f charges having been drained is defined in Equ. 3.2, where α_N and $\alpha_C(f)$ depend on the recovery capability behavior of the battery in the considered recovery phase. A small α_N value corresponds to a low recovery capability which means a high battery internal resistance for draining more charge units. The authors in [102] assume α_N to be constant and $\alpha_C(.)$ as a piecewise constant function of the number of charge units having already been extracted from the electrolyte.

$$p_j(f) = q_0 Exp^{(N-J)\alpha_N - \alpha_C(f)} \tag{3.2}$$

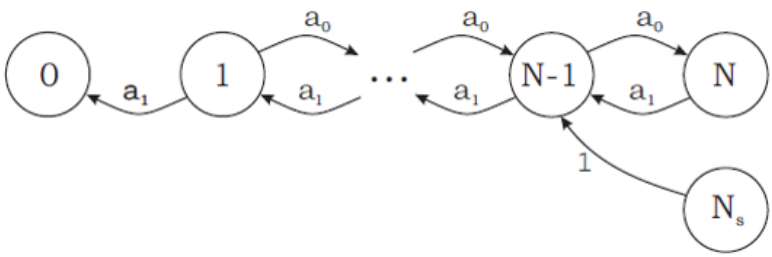

Figure 3.8: The binary pulsed discharge model proposed by Chiasserini and Rao in [102]

For all the models cited above, several extensions have been introduced in the original stochastic model to improve its quality, e.g. some models have considered the recovery rate both state and phase dependent.

The results obtained by the stochastic battery model can be close to those of the electrochemical model with a maximum deviation of 4% and an average deviation of 1% [94]. These results show also that the stochastic model gives a good qualitative description of the battery behavior under pulsed discharge. That is why we choose to use this model in the rest of this thesis work.

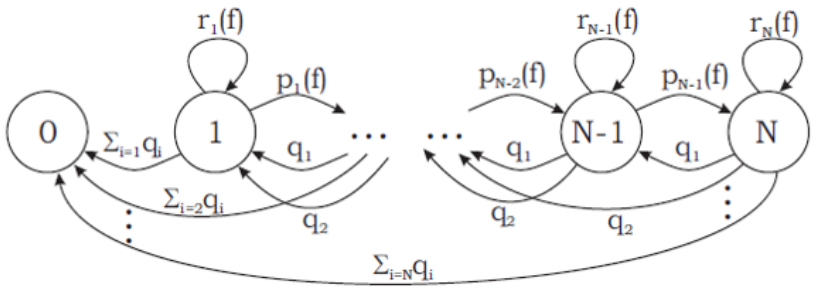

Figure 3.9: The generalized pulsed discharge model proposed by Chiasserini and Rao in [102]

3.3 The recovery effect of batteries

In order to exploit effectively all the available battery charge, the recovery effect of chemical batteries is exploited in this section. This effect will lead to extract more power from the batteries when idle times among usage cycles are implemented [103].

3.3.1 Recovery effect definition

The recovery effect is a phenomenon observed in batteries. This electrochemical effect allows more power to be extracted from the battery when periods of inactivity are implemented in the operating cycle of the embedded system. This phenomenon has been used to develop various power management techniques as in [104, 105].

In fact, by scheduling a period of inactivity during operation, the usable capacity of the battery can increase thanks to an automatic regeneration mechanism that occurs inside the battery using the active materials remaining in the electrolyte during these periods of inactivity [101]. This mechanism is called the recovery effect and can be modeled as a Markov chain model to describe the battery discharge process.

the concept of recovery effect is described in Figure 3.10. Battery voltage recover during the idle periods which takes the battery far away from the vicinity to the threshold voltage. When the battery supply voltage falls below a certain voltage threshold, the battery is considered to be completely discharged and the embedded system can no longer function.

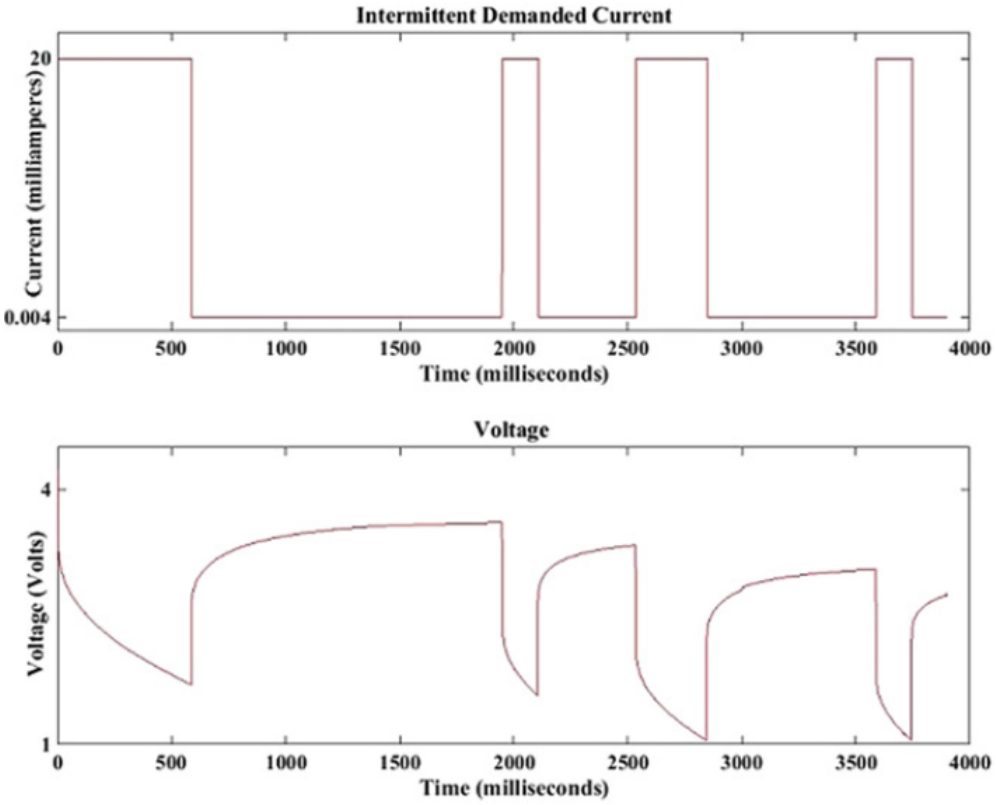

Figure 3.10: Illustration of the recovery effect concept from [104]

3.3.2 Battery recovery-aware features and background

Regain of the battery voltage can be noticed, when no load current is provided. It is caused by what we call the recovery effect that many works in the field of wireless communication networks and embedded systems have considered the existence [105, 106, 107, 108]. In fact, intermittent discharge profile can increase the usable capacity inside the battery since their cells are able to replenish themselves by recovering charge units during the idle periods of time [81].

Therefore, we consider a pulsed current discharge with in-between idle periods as described in [101]. In this situation, a charge recovery phenomenon occurs inside the battery when no current is drained outside, e.g. in the idle periods of a duty cycle mode. In those periods, the battery can replace the consumed charge units by those being extracted from the electrolyte solution according to a diffusion process because the gradient of the active materials decreases from the electrode towards the electrolyte. Consequently, new charge

units become available for future energy consumption cycles [81]. To exploit this effect we need first to model the battery. Actually, many battery models are proposed in literature. Those models can be classified into four main categories [94] that are electrochemical, Electrical circuit, Analytical and stochastic models.

For this chapter, we consider the stochastic model for the battery state behavior as described in [100]. We only consider the recovery effect and neglect all other phenomena occurring inside the battery. According to this model, the battery behavior follows a Markovian process evolution starting from a fully battery charge state and stopping when either the whole theoretical capacity is consumed or the zero charge state is reached. This latter is considered to be a trapping state where the battery becomes out of use.

As acknowledged, the wireless communication is the biggest source of energy expenditure in WSNs. We assume that the major source of energy consumption is due to the data transmission process and will neglect the consumption of all other sensor radio modes such as receiving, idle listening or sleeping ones.

Starting from a fully battery charged state, we assume that the decision making, for transmitting or not a data packet, follows a Bernoulli process with probability $a_1 = q$ that a data packet will be transmitted. Hence, in each time slot, either the transmission will occur with probability $a_1 = q$ or the data packet will be dropped with probability $a_0 = 1 - q$. The system state 0 is a trapping state and once this state is reached the battery is declared dead. The state B_{max} is the upper-bound state and the limit of the battery charge, it can be visited more than once due to the charge recovery process. The obtained model for the battery charge and discharge mechanisms is summarized in Fig. 3.11 where a dummy state B_{start} is added to show the starting time [81].

3.3.3 Experimental proofs of battery recovery effect

According to experimental results as in [109], both for TelosB and Imote2 intermittent discharge profile the battery runtime can be enhanced with a gain up to 25% and 38% respectively. For TelosB sensors, Figures 3.12 and 3.13 give the normalised battery runtime extent when a discharge battery current profile is considered and the acheived gain of normalised runtime respectively. The same thing is given in Figures 3.14 and 3.15 for

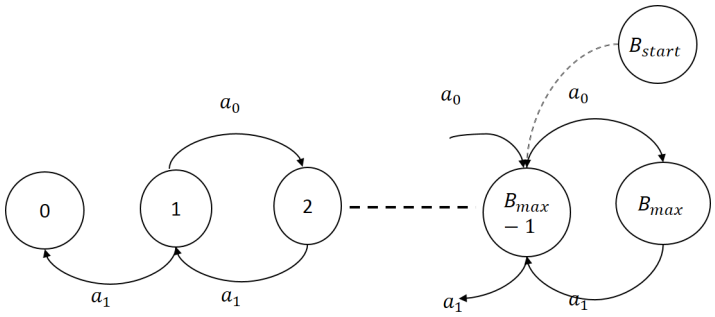

Figure 3.11: Binary battery operating model with Bernoulli process probability

Imote2 sensors.

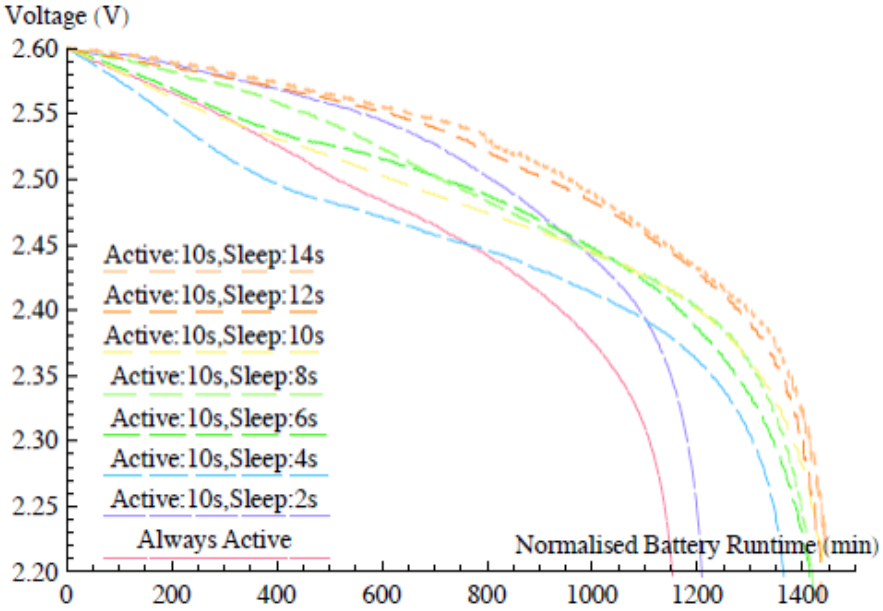

Figure 3.12: Battery voltage discharge of TelosB w.r.t different sleep time durations [109]

With intermittent current discharge, batteries last longer in comparison with a continuous battery discharge current profile. According to the Figures 3.12, 3.13, 3.14 and 3.15 clear signs of battery recovery are shown. For the same activation period, long runtime duration result from long sleep time periods in a non-linear form as depicted in the given figures. Even though the embedded system energy consumption still exist during the idle periods due to the inner timer and other processing activities (e.g., in sleep mode, TelosB consumes $6,1\mu A$ and Imote2 consumes $0,38mA$), the recovery still producing its effect even in case of low battery consumption [109].

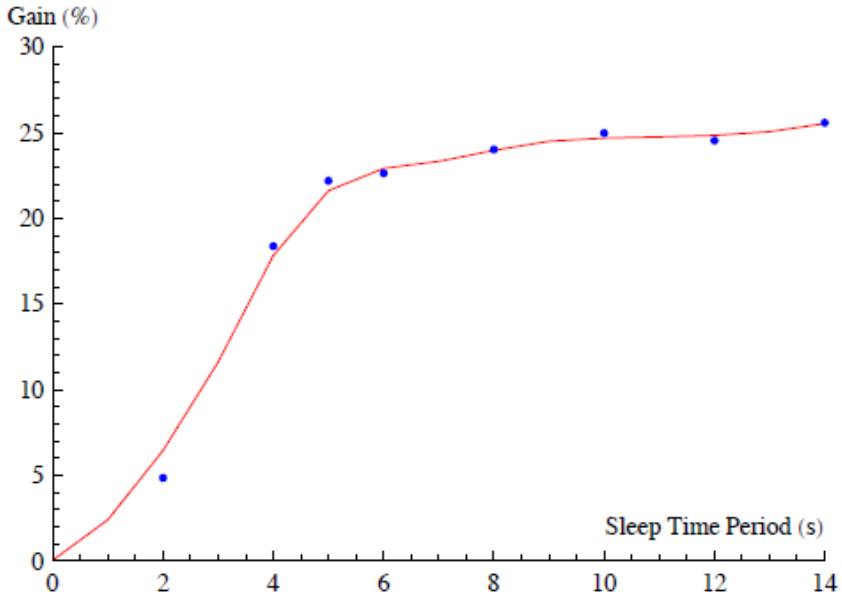

Figure 3.13: Normalised battery runtime gain for TelosB as compared to the continuous discharge mode [109]

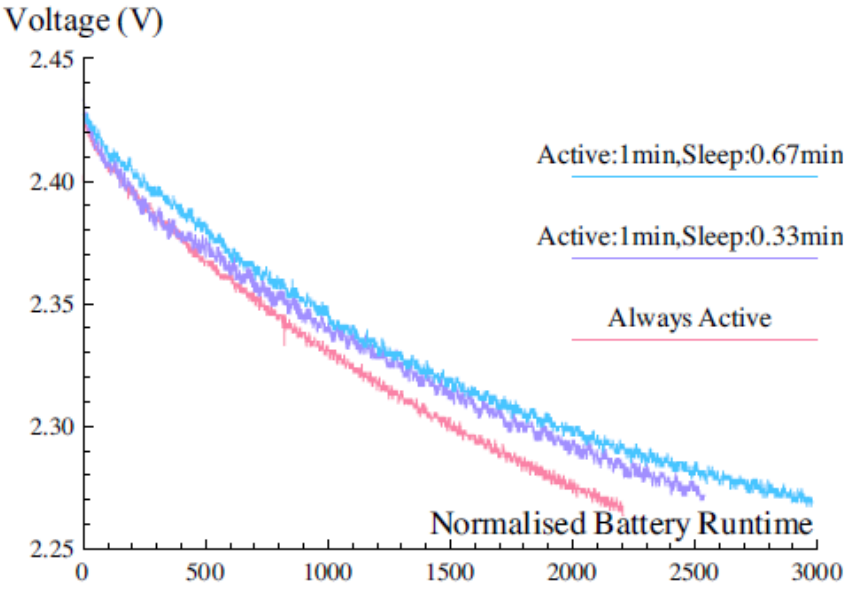

Figure 3.14: Battery voltage discharge of Immote2 w.r.t different sleep time durations [109]

In order to make the best use of the characteristics both of primary or secondary batteries, we first try to find the battery model that will put the most emphasis on the essential qualitative aspects reflected by the use of energy harvesters or the exploitation of the

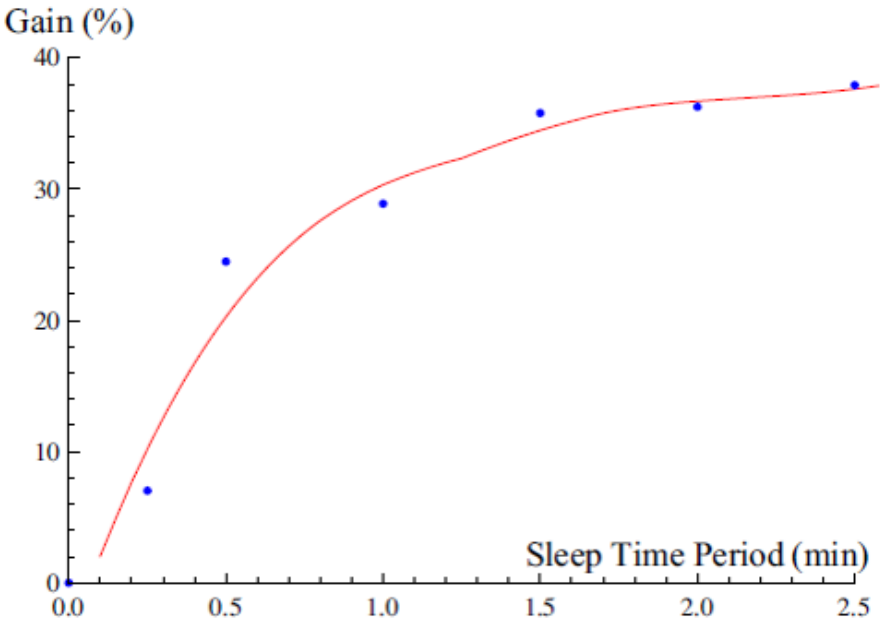

Figure 3.15: Normalised battery runtime gain for Imote2 as compared to the continuous discharge mode [109]

recovery effect on the communication performance of the embedded systems.

3.4 Energy harvesting sources for battery charging

Energy harvesting is a technique that captures and recovers energy from the environment as a source of energy and converts it into electrical energy [110]. Common sources of ambient energy for energy harvesting are mechanical energy from vibration and deformation, thermal energy from furnaces and other heating sources, solar energy from all forms of light sources such as lighting and the sun, electromagnetic energy captured by inductors and transformers, wind energy and fluid energy resulting from the circulation of air and liquids, human energy generated by the movements of the foot, skin and blood, and chemical energy from biological processes [111]. Here are some examples of energy harvesting devices and their applications:

- Harvesting thermal energy from machine heat to extend the life of a wireless sensor [112];

- Recovering energy from mono-crystalline solar cells to power wireless sensors used in indoor applications [113];
- Energy harvesting by inductive coupling to radio frequencies (RF) adapted to a device implanted in the human body [114];
- Wind energy recovery in distant regions for low power autonomous sensors [115].

The classification of energy harvesting can be organized according to the form of energy used to recover electrical power. For example piezoelectric harvesting devices convert mechanical energy into usable electrical energy. Some of the categories of energy collection sources are thermoelectric and photovoltaic cells, wind turbines and mechanical vibration generators, and devices such as piezoelectric devices, electromagnetic devices, etc [116]. The power production densities for each harvesting method are shown in Table 3.1 [117].

Harvesting technology	***Power density***
Solar cells (outdoors at noon)	15 mW/cm^2
Piezoelectric devices (shoe inserts)	330 $\mu W/cm^3$
Thermoelectric cells (10°C gradient)	40 $\mu W/cm^3$

Table 3.1: Power densities for different energy harvesting sources

Numerous studies have been carried out on the recovery of energy from the environment to convert it into electrical energy [118]. Certain properties must be taken into account to characterize an energy harvester. Fry et al in [119] stated that a special care should be taken while using energy harvesters in embedded systems by considering different types of criteria such as power density, size, shape and wight, operating temperature range, water resistance and maintenance properties. The order of magnitude of the energy recovered is often insufficient to meet the needs of on-board systems, so improving the amount of energy recovered remains a major challenge. The most widely used harvesting principles and technologies, which will be described below, are piezoelectricity, photovoltaic cells and thermoelectricity.

3.4.1 Piezoelectric energy generators

Piezoelectricity is the most exploited effect in the field of mechanical vibration source of energy. The piezoelectric effect produces low voltages when the crystals are deformed mechanically. Mechanical pressure can come from many different sources such as human movement, earthquakes, acoustic noise or vibrations of rotating machines. Marzencki et al in [120] presented a new ambient energy recovery system which converts the energy of environmental mechanical vibrations into electrical energy useful for powering wireless sensors. The diagram in Fig. 3.16 shows a system that uses a piezoelectric generator (transducer) based on micro-electromechanical systems (MEMS), a custom ASIC for voltage rectification and a super-capacitor for energy storage [120]. For example, the micro-electromechanical devices developed by the company Microgen (USA) captures the energy of the ambient vibrations by means of a small pendulum that provides electrical energy by piezoelectric effect as shown in Fig. 3.17 [121]. It should be remembered that piezoelectric solutions can be very largely miniaturized as shown in Fig. 3.18, where this very small harvester can generate more than 200 μW of power when exposed to a vibration amplitude of 1.5 g. The energy collected is processed by an integrated circuit to charge an ultra-capacitor at 1.85 volts.

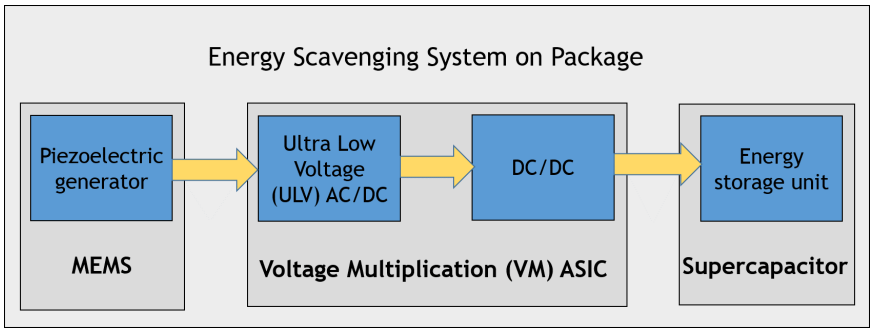

Figure 3.16: Schematic of the energy scavenging system

3.4.2 Photovoltaic cell sources

Solar radiation is the most obvious energy available and it is logically in this field that very active research is being developed. The yield of conventional cells doesn't really allow

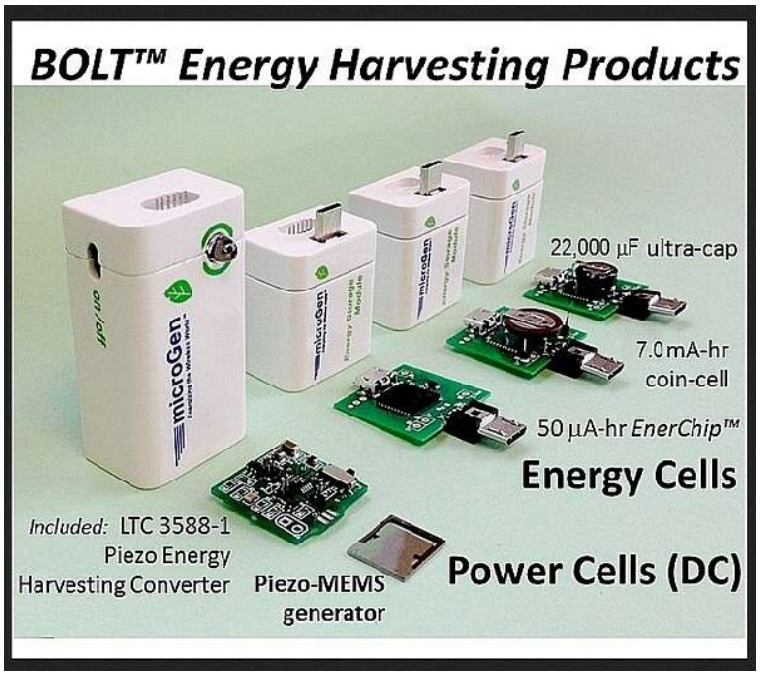

Figure 3.17: From left to right - (1) $BOLT^{TM}$ Power Cell; (2) piezo-MEMS energy harvesting generator or MPG; (3) ESM-B1 using 50 $\mu A.hr$ solid state battery (Cymbet Corp); (4) ESM-B2 using 7.0 $mA.hr$ rechargeable coin cell; (5) ESM-C using 22 mF ultra-Capacitor. Source : Micogen

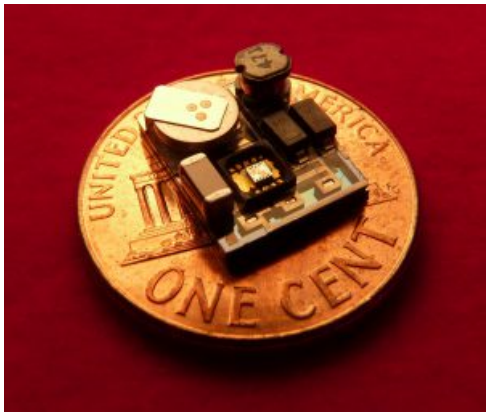

Figure 3.18: Piezoelectric device developed in 2011 at Michigan university. The active part of the harvester occupies a volume 27 mm^3 and the obtained power is 200 μW at $1.85V$. Source : Michigan University

miniaturization but recent developments have enabled the production of high sensitivity sensors usable especially in low light conditions. For example, solutions that rely on small

Photovoltaic cell module to supply a sensor node were proposed by Prometheus [122] and Heliomote [123] as shown in Fig. 3.19 and Fig. 3.20 respectively.

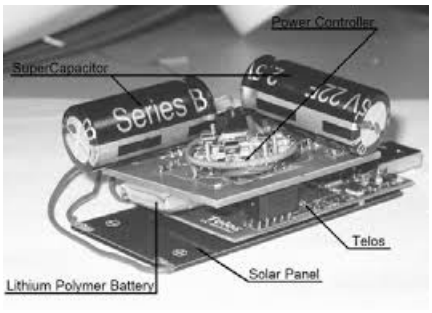

Figure 3.19: Prometheus: Perpetual Self powered Telos Mote with solar energy harvesting system. Source:[122]

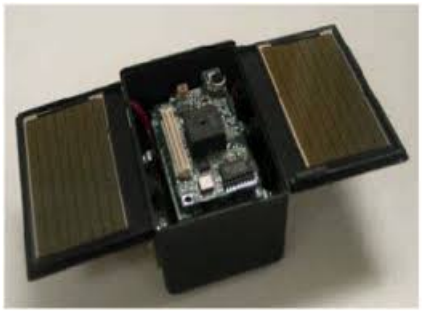

Figure 3.20: Heliomote: The solar energy harvesting sensor node. Source:[123]

Photovoltaic energy is certainly the most developed and most used for a large power scale. A new generation of broad spectrum sensors for low power systems is capable of using outdoor and indoor light energy. A wide voltage scale is possible because these generators can be connected both in series and in parallel. If energy recovery by photovoltaic effect is a technique now quite old, its use from dim lights or artificial lights is recent. Since the development of dye-sensitive solar cell (DSSC) technology, huge improvements have been made. For example, GaAs solar cells that was then developed produces a power density 3x greater than the DSSC cell modules. As order of magnitude, a credit card-sized GaAs cell can provide up to 0.67 mW or 4 mW to a wireless sensor node in a dimly lit hallway ($\approx$ 200 lux) or a well-lit office space ($\approx$ 1000 lux) respectively as depicted in Fig. 3.21 [124].

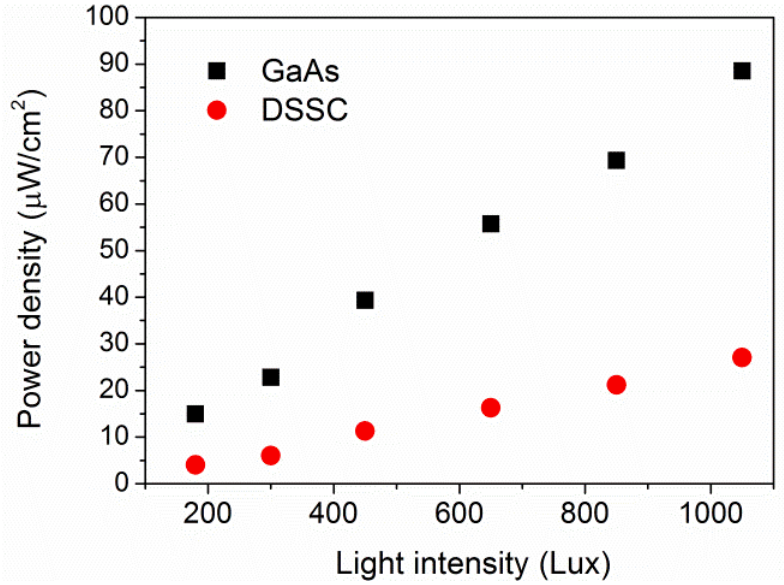

Figure 3.21: Comparison between power density of GaAs solar cells and DSSC module under low light conditions. Source:[124]

3.4.3 Thermoelectric generator sources

Thermoelectric technologies use the Seebeck effect, i.e. the possibility of generating electricity from a temperature gradient. Actually, the Seebeck effect uses the notion of thermoelectric couple in which charge carriers, such as electrons and holes, move between the cold source and the hot source, under the action of the thermal gradient. The materials to produce such an effect must have a high electrical conduction and low thermal conduction. The maximum efficiency of a thermoelectric material is determined by its thermoelectric figure of merit ZT as given in Equ. 3.3, where σ the electrical resistivity, κ thermal conductivity and S the Seebeck coefficient and T is the absolute temperature [125]. Three materials are commonly used in that field which are the Bismuth telluride ($B_{i_2}T_{e_3}$) for low-temperature (from near home temperature up to $\approx 473^oK$), the Lead telluride (P_bT_e) for mid-range temperature ($500 - 900^oK$) and the Silicon germanium (S_iG_e) for high-temperature ($> 900^oK$) as depicted in Fig. 3.22.

$$ZT = \frac{S^2}{\sigma \times \kappa} \times T \tag{3.3}$$

The thermoelectric generator module are built with an array of p-n junctions that are

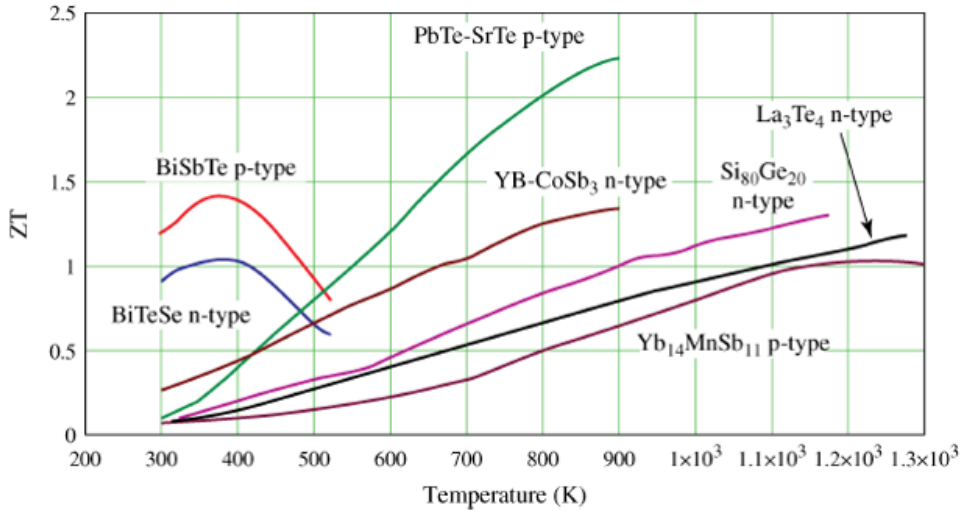

Figure 3.22: Figures of merit for various thermoelectric materials. Source : [126]

electrically connected in series and thermally connected in parallel in order to have the desired electrical current and voltage. The p-n junctions are placed between two parallel ceramic plates. The plates serve as a mounting surface while providing structural rigidity and electrical insulation against short circuits as shown in Fig. 3.23.

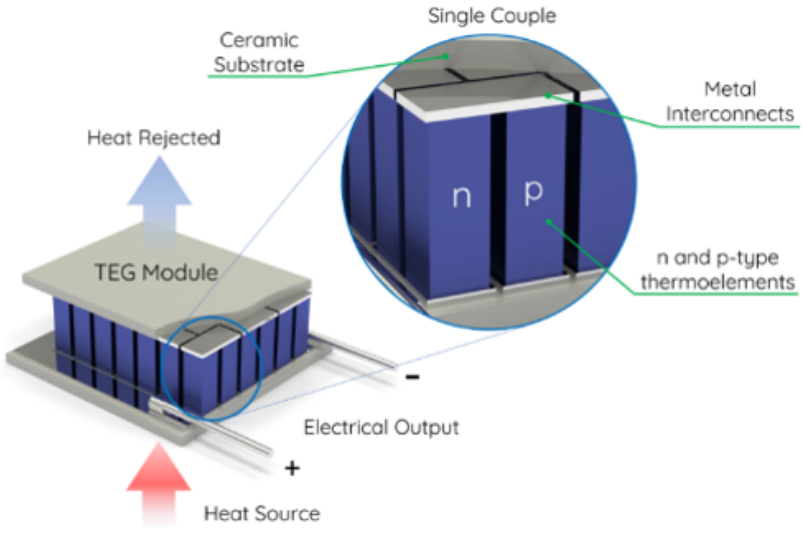

Figure 3.23: Thermoelectric generator (TEG) module. Source : [127]

Although their energy efficiency is not very high, the thermoelectric generator are indispensable in some existing niche markets, in particular space probes, e.g. Cassini-Huygens

probe and Curiosity robot [128]. Figures 3.24 and 3.25 show some example of thermoelectric generator modules proposed by two of the companies specialized in this field: II-VI Marlow (USA) and Micropelt(Germany) respectively.

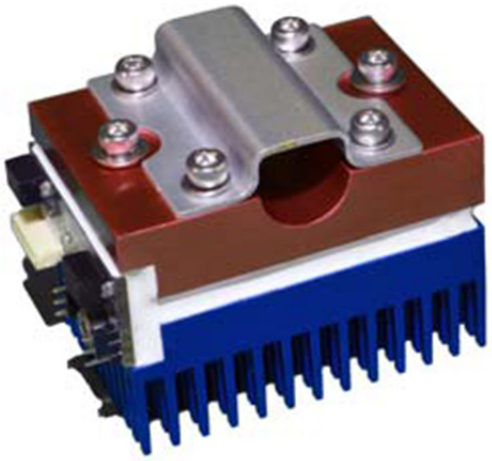

Figure 3.24: Thermal Energy Harvesting adapted to temperature gradients ranging from 5.5 to 60°C - Power: 0.3 mW for $\Delta t = 10°C$. Source : Marlow

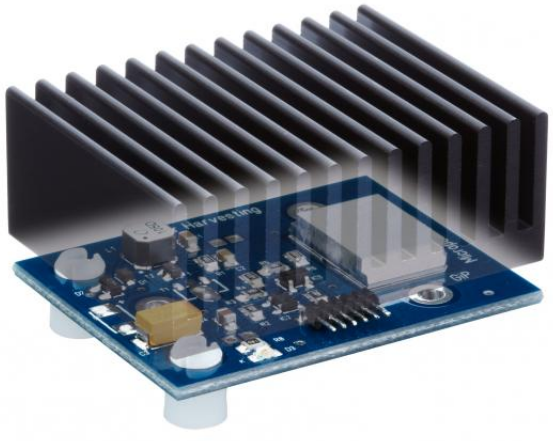

Figure 3.25: Micropelt's TE-CORE is a maintenance-free self-sustaining power source for ultra-low power wireless applications. Source : Micropelt

3.5 Conclusion

In this chapter, we have presented a bibliographical study of battery modeling in order to identify the model best adapted to the needs of this thesis. First of all, after an introduction on the description of batteries for embedded systems, we presented the main characteristics of battery behavior, including the definitions of their theoretical and nominal capacities that will be exploited for the problem of energy optimization in Chapters 4 and 5. Energy harvesting techniques and the recovery effect were also presented in order be exploited later for calculations of optimal transmission strategies. Battery modeling was also discussed in its five most popular forms and we concluded that the model that best meets the needs of our work is the stochastic model. The stochastic model we will use for the rest of this thesis is based on the Markovian representation of the battery behavior, which will allow the use of algorithmic techniques such as dynamic programming and reinforcement learning to solve any energy optimization problem in the context of embedded systems communications.

Chapter

4 Battery-powered embedded systems with energy harvesting capabilities

In this chapter, we consider the case of point-to-point wireless communication for embedded systems that are supposed to be battery-powered and equipped with an energy harvester. Depending on the battery level and the amount of energy collected during the current time slot, the transmitter makes the decision whether or not to transmit data packets according to a given policy in order to maximize its reward over the operating period. We will first consider that a prior stochastic knowledge of all transition probability matrices is available in order to be able to find the optimal policy using an algorithm derived from the dynamic programming methods. In the absence of such stochastic knowledge, this time we will adopt algorithms from Reinforcement Learning methods to learn and estimate optimal policies. Finally, the results obtained will then be discussed and compared.

4.1 Introduction

For this chapter, we consider the system model in Fig. 4.1 which can be described as self-powered due to its energy harvesting and management capabilities. In fact, many application-oriented research institutes are carrying out projects in this field and may be interested in a new generation of independent and self-managed energy sensors. By way of illustration, the project "Daedalus modular, energy-independent tracking systems" led by

the Fraunhofer Institute for Integrated Circuits (IIS, Germany) exploits solutions related to the development of intelligent and efficient self-powered sensors [129].

Two scenarios are considered based on the information available for the calculation of optimal policies. In the first scenario, it is assumed that stochastic knowledge of the behavior of the environment is already available. The optimal policy can then either be calculated in each sensor, or disseminated by the gateway to each of them so that they can make an optimal decision at the transmitter level based on the system states they are facing. In the second scenario, no information is available on the stochastic behavior of the operational environment, so policies are drawn from the series of decisions made and improved sequentially from one episode to the next during a preliminary learning sequence.

For the purposes of the simulation, we assume that the data packets arrive at the transmitter as a time slot timing occurrence. At the beginning of each time slot (T_S), a data packet is supposed to be received. If this packet is not transmitted in the next time slot, it is automatically destroyed. The harvested energy that arrives within a time slot is stocked if and only if the battery is not full, otherwise this energy is lost. We also consider that the data transmission operation is the main and dominant cause of energy consumption. We assume that the state of the wireless channel does not change for the duration of a time slot but may vary from one time slot to another. We consider that the data packet arrivals, the energy harvested units and the wireless channel attenuation levels can be all modeled as Markov processes as in [130].

The main objective of the transmitter unit within the sensor is to maximize the cumulative data transmitted to the destination during its operating time, taking into account the constraints of energy availability. For this purpose, we provide a complete analysis of the optimal communication system based on a stochastic model of the system. We present a computing methods to obtain an optimal policy according to the availability of information. The information being considered are the transition matrix probabilities of data sizes, energy harvesting amounts and the channel states. In this work, both model-based and the model-free approaches will be treated. So, the main contributions of this chapter can be summarized as follows:

- Based on the models of different components of the system studied, we provide a complete analysis to obtain the optimal policy for a maximum expected cumulative transmitted data.
- We compare through the numerical results obtained the performance achieved by our optimal policies in terms of expected cumulative transmitted data.

The rest of this chapter will be organized as following: Section 2 will be dedicated to the system modeling and the presentation of the optimization problem, Section 3 is devoted for computing methods to obtain the optimal policies in model-based case study where sufficient knowledge is provided in advance, in Section 4 the same objective as in the previous section is carried out in case of model-free optimization problem where no prior information are available, the obtained results are presented and compared in Section 5 using numerical analysis and matlab simulation, and finally Section 6 will conclude this chapter.

4.2 System modeling

In this work, we consider a wireless system with a transceiver powered by a battery and equipped with external energy harvesting device. We assume that the battery has a limited storage capacity and will often be replenished by the energy harvester as in [131]. We assume also that our system operates according to a time-slotted fashion with fixed duration periods of equal values T_S. The data arrives in packets at each time slot T_S with different sizes. The environmental energy is harvested with different amounts within each time slot. We consider that the communication channel takes a constant state over each T_S and we assume that this state changes only from one T_S to the next one.

We consider that a received data packet will be transmitted over the next T_S duration otherwise it will be destroyed. The energy harvested during a giving time slot T_{S_n} serve for the replenishment of the battery and is assumed to be ready for use in the following time slot $T_{S_{n+1}}$.

Our main objective is to perform a good use of the energy available in the battery with the purpose of maximizing the expected total data transmitted during the whole operating

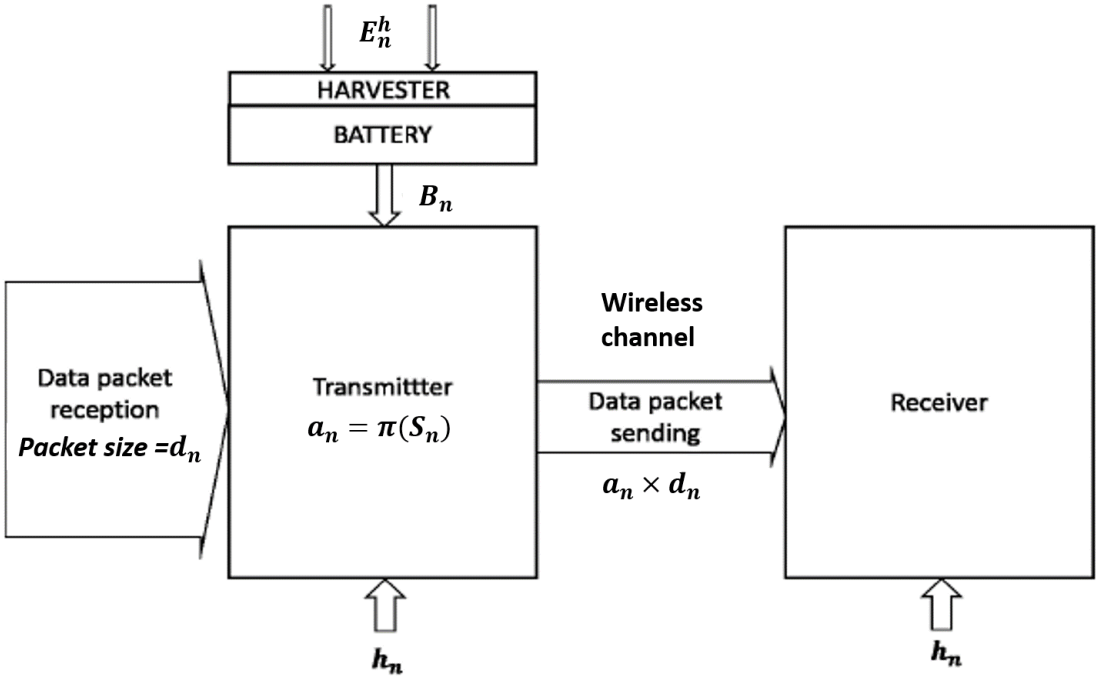

Figure 4.1: Battery-powered system with energy harvesting

time of the system in association with a discounted reward process in terms of transmitted data according to a discount factor rate γ.

We modelize the data packet sizes arriving during the current time slot T_S as a stochastic process characterized by a first order Markov model. We made the choice to consider that the data packet sizes arriving during T_{S_n} is d_n that belongs to the set of all possible data packet sizes such as $\{d_1, d_2, \ldots, d_{N_D}\}$, where N_D is the total number of all possible packet sizes considered. We note that the transition probability that the considered data packet size shifts from one packet size d_j to d_k within one time slot T_S as $p_d(d_j, d_k)$.

In addition, we assume that the amount of harvested energy during T_{S_n} follows also a stochastic process characterized by a first order Markov model. We consider that the harvested energy during T_{Sn} is E_n^h that belongs to the set of all possible harvested energy amounts such as $\{e_1, e_2, \ldots, e_{N_E}\}$, where N_E is the total number of all possible harvested energy amounts considered. We note that the transition probability that the considered harvested energy amount shifts from one amount e_j to e_k within one time slot T_S as $p_e(e_j, e_k)$.

Also, we consider the channel attenuation level evolution from one time slot T_S to the next one as a stochastic process characterized by a first order Markov model. We make the choice to consider that the channel attenuation level during T_{S_n} is h_n that belongs to the set of all possible channel states such as $\{h_1, h_2, ... ,h_{N_H}\}$, where N_H is the total number of all possible channel states considered. We note that the transition probability that the channel state switches from one channel state h_j to h_k within one time slot T_S as $p_h(h_j,h_k)$.

We note, $E_n^{T_S}$ the amount of energy being spent when the transmitter is performing a data packet transmission of size d_n over the channel being in state h_n towards a receiver, no matter whether the transmission is being performed successfully or not. And we note, e the elementary amount of energy (i.e. a charge unit) which is equivalent to the minimum of energy needed to transmit a data packet of minimal size across the channel that presents the minimal attenuation.

We consider that a transmission is performed successfully if the $E_n^{T_S}$ amount of energy consumed is equal to the right quantity of energy needed in terms of charge units according to both d_n and h_n values. It appears trivial that for long data packet sizes and/or bad channel conditions we would need more than one charge unit to perform a successful transmission.

We assume that all the information related to the battery charge level, the energy being harvested, the size of the arrived data packet, the current channel state are known at the transmitter side, and so are the information related to the channel state during the considered time slot that are known at the receiver side.

At the beginning of each T_S the transmitter makes the decision, according to the information available concerning the remained battery charge, the data packet size and the channel attenuation level, of either to transmit the incoming packet or to drop it when following a designed policy or according to a learning process. We consider a_n as an indicator function of the decision being made by the transmitter whether to transmit tha data ($a_n = 1$) or to drop it ($a_n = 0$). a_n is belonging to the set of values $A = \{0, 1\}$. The transmitter makes the transmission decision and fixes the power to be used at the beginning of each time slot and cannot be changed therefore.

Some constraints have to be taken into account to insure a good achievement of the optimization objectives as given in Equ. 4.1 and 4.2.

$$a_n E_n^{T_S} \leq B_n, \tag{4.1}$$

$$B_{n+1} = \min\left\{B_n - a_n E_n^{T_S} + E_n^h, B_{max}\right\} \tag{4.2}$$

where B_n is the current battery charge, B_{max} the maximal charge capacity of the battery. After each iteration, the battery state is then updated to the next state according to energy spent for the transmission and the harvested energy received and stocked.
The main goal of the optimization problem is to maximize the expected total transmitted data over the whole operating time which will be equivalent to the total activation time duration of the transmitter. The objective of this optimization will be described in Equ. 4.3 and 4.4.

$$\max_{\{a_i\}_{i=1}^{\infty}} \sum_{n=1}^{\infty} \gamma^{n-1} a_n \times d_n \tag{4.3}$$

$$\text{subject to (4.1) and (4.2)} \tag{4.4}$$

Our system will be then approximated by a finite-state discrete-time MDP model [132]. The MDP model will constitute the basic platform for the decision making giving the randomness nature of the states taken by the system and its environment that will be considered fully observable so that the current state completely characterizes the process. Our considered system state is composed of four elements that are E_n^h representing the quantity of energy that would be harvested during the current time slot, d_n the data size packet received in the previous time slot T_{Sn-1} and being ready to be transmitted during the current time slot, h_n the actual attenuation of the channel and B_n the available capacity charge in the battery at the beginning of the time slot, so S_n is $\langle E_n^h, d_n, h_n, B_n \rangle$. At each period T_{S_n}, the system state is S_n. The components of the system state are then discrete, and the set of all the possible system state S is of a finite number of elements. S is denoted by $S = \{S_1, S_2, ..., S_{N_S}\}$ where N_S is the total number of all the possible system states obtained by combining all the component values of the three dimensional S_i element.

A is the set of all possible actions that the agent can take, either to transmit the data packet or to drop it. Therefore, A is equal to $[0, 1]$.

A policy, denoted by π, will defines the behavior of the transmitter at each time according to the evolution of the system state S_n, i.e. it is the linking map between the set of the observed system state and the set of the actions to be taken by the transmitter as a decision maker agent.

The reward function considered represents the goal of the transmitter as an agent, i.e. it translates the pair vector composed of the action applied to the system and the current state being taken to a single reward number. In our case, the reward represents the quantity of bits being transmitted when evolving from one system state to the next according to the decision made by the transmitter and is denoted by $R_n = a_n \times d_n$.

The main transmitter objective is to maximize the sum of the resulting rewards in the long run of the system considering the discounted aspect of the data being transmitted. This objective is addressed by the state-value function $v_\pi(S_i)$ of the MDP system which represents the expected total discounted amount of rewards when starting from state S_i, and then following the policy π. The state value function that will be our main tool for solving the resulting MDP system is defined as follows :

$$v_\pi(S_i) \triangleq \sum_{\forall S_k \in S} p_{\pi_{(S_i)}}(S_i, S_k)[R_{\pi_{(S_i)}}(S_i, S_k) + \gamma v_\pi(S_k)] \tag{4.5}$$

The state-value function $v_\pi(S_n)$ is then expressed as a combination of the expected immediate reward and the next state-value function denoted by $v_\pi(S_{n+1})$.

In the case of a learning policy, the transmitter objective becomes to maximize the reward in long run interaction with the environment on the basis of his own experience. To act like this, the transmitter has to consider the value of the action-state value function in each step when starting from state S_i, taking action a_l and then following policy π. The action-state value function is defined as follows :

$$q_\pi(S_i, a_l) \triangleq \sum_{\forall S_k \in S} p_{a_l}(S_i, S_k)[R_{a_l}(S_i, S_k) + \gamma v_\pi(S_k)] \tag{4.6}$$

The state-action value function $q_\pi(S_n, a_n)$, when starting fron state S_n, taking action a_n and then following policy π, is by the same way expressed as a combinaition of the

expected immediate reward and the next state-value function denoted by $v_\pi(S_{n+1})$.
We can say that a policy $\pi^{'}$ is better than or equal to policy π, denoted by $\pi^{'} \geq \pi$, if and only if the expected total discount amount of rewards obtained following policy $\pi^{'}$ is greater than or equal to that when following policy π which means that $v_{\pi'}(S_i) \geq v_\pi(S_i)$. The optimal policy π^* is defined as the policy which is better or equal to any other possible policy conducting to the optimal state value functions $v_{\pi^*}(S_i)$ with i=1, 2, ...,N_S. So is the case of the action state value $q_{\pi^*}(S_i, a_i)$ being greater or equal to any action state value following other policies strategies. The optimal state value is linked to the optimal action state value by the following relationship:

$$v_{\pi^*}(S_i) = \max_{a_i \in A} q_{\pi^*}(S_i, a_i) \tag{4.7}$$

According to the system state S_i, the optimal policy is then greedy over the set the action state values of all possible pairs (S_i, a_i). That is, the problem to solve is to choose the action the transmitter should take to maximize his rewards and then follows the policy π. To reach to optimal expected total reward one should consider that the action state value is the combination of the maximum immediate reward obtained by choosing the action to take according to the greedy policy and the maximum action state value for the next state so that:

$$q_{\pi^*}(S_i, a_l) = \sum_{\forall S_k \in S} p_{a_l}(S_i, S_k)[R_{a_l}(S_i, S_k) + \gamma \max_{a_i \in A} q_{\pi^*}(S_i, a_i)] \tag{4.8}$$

Two approaches will be taken into account in our study in order to solve the resulting MDP problem depending on whether we manage to know the system model so that we are dealing with a model based Markovian decision problem, otherwise we are in the situation of a model free control problem case.
In a model-based configuration, we assume that we have good knowledge of the stochastic behavior of the model with prior information on the different components of the MDP model, as it can be described by a 5 tuple $\langle S, A, P, R, \gamma \rangle$, where S is the set of states, A is the set of actions, P the state transition probability matrix, R the reward function and γ a discounted factor as in [132]. In that case of study, we have definitely sufficient knowledge of the transition probability matrix, the set of actions and states and the

associated generated rewards according to each state-action pairs being considered to be able to apply DP algorithms in order to find optimal transmission policy π^*.

Concerning the case of a model free control, we assume that we don't have any prior stochastic information about the system being studied. We manage then to permit the transmitter to explore the environment behavior itself and exploit the set of information resulting from experiences by following a reinforcement learning approach. The main objective is to learn how to make right decisions to tend towards an approximation of the policy π^* that should maximize the total rewards accumulated over time.

4.3 Model-based control optimal policy design

In case of a model based, the transmitter care mainly about performing successful and efficient transmission of the incoming packets. We employ algorithms derived from the dynamic programming field (DP) to solve the MDP problem resulting in Equation 4.4 [133]. Both Policy iteration algorithm (PI) and Value Iteration algorithm (VI) will be used.

Given that our system MDP model has finite action and state spaces and bounded and stationary reward function, both PI and VI are proven to converge to the optimal policy when the discount factor is $0 \leq \gamma < 1$. The solution considered at this stage is to combine the set of equation composed of (4.5 , 4.6, 4.7 and 4.8) to obtain the optimal policy.

The key idea of PI and VI algorithms is to use the value functions to structure and organize the search of the optimal policy that guarantees the maximization of the achieved cumulative reward. Both VI and PI operate by exploiting Equation 4.9.

$$\begin{aligned} v_{\pi_l}(S_i) &= \max_{a_i \in A} q_{\pi_l}(S_i, a_i) \\ &= \max_{a_i \in A} \sum_{\forall S_k \in S} p_{a_l}(S_i, S_k)[R_{a_l}(S_i, S_k) + \\ &\gamma \max_{a_i \in A} q_{\pi_{l-1}}(S_i, a_i)] \end{aligned} \tag{4.9}$$

For all $S_i \in S$ and for arbitrary v_{π_0} initialization, the sequence $\langle v_{\pi_0}, v_{\pi_1}, v_{\pi_2}, ..., v_{\pi_k}, ... \rangle$ converges to v_{π^*} under the same conditions that guarantees the existence of v_{π^*} itself.

The algorithms that have been implemented in our system decision making according to PI and VI algorithms are described in Algorithms 7 and 8. In fact, both PI and VI algorithms are proven to converge to the optimal policy for the discounted finite MDP being considered [64].

Algorithm 7 *Policy Iteration Algorithm*

1.Initialization

Choose $v_\pi(S_i) \in \mathbb{R}$ and $\pi(S_i) \in A$ arbitrarily for all $S_i \in S$

Initialize the array of $v_\pi(S_i)$, for example put $v_\pi(S_i) = 0$ for all states $S_i \in S$

2.Policy Evaluation

Repeat

$\Delta \leftarrow 0$

for each state $S_i \in S$ **do**

$temp \leftarrow v_\pi(S_i)$

$v_\pi(S_i) \leftarrow \max\limits_{a_i \in A} \sum\limits_{\forall S_k \in S} p_{a_i}(S_i, S_k)[R_{a_i}(S_i, S_k) + \gamma v_\pi(S_k)]$

$\Delta \leftarrow \max(\Delta, |temp - v_\pi(S_i)|)$

end for

Until $\Delta < \theta$ (a small positive number)

3.Policy Improvement

$policy - stable \leftarrow true$

for each state $S_i \in S$ **do**

$temp \leftarrow \pi(S_i)$

$\pi(S_i) \leftarrow \operatorname{argmax}\limits_{a_i \in A} \sum\limits_{\forall S_k \in S} p_{a_i}(S_i, S_k)[R_{a_i}(S_i, S_k) + \gamma v_\pi(S_k)]$

If $temp \neq \pi(s)$, then $policy - stable \leftarrow$ false

end for

If $policy - stable$, then stop and return v_π and π; else go to 2

In our particular model, the computation complexity of calculation is bounded by $O(2^{N_S}/N_S)$ and $O(2{N_S}^2)$ in the PI and VI cases respectively [134]. Therefore, VI is considered to be more faster that PI. Thus, we will only adopt the VI algorithm for the next chapter.

Algorithm 8 Value Iteration

1. Initialization step:

for each state $S_i \in S$ **do**

Initialize $v_\pi(S_i)$ arbitrarily

end for

2. Value Iteration:

$\Delta \leftarrow 1$

$\epsilon \leftarrow 0.001$

while $\Delta > \epsilon$ **do**

$\Delta \leftarrow 0$

$v \leftarrow v_\pi$

for each state $S_i \in S$ **do**

$$v_\pi(S_i) \leftarrow \max_{a_i \in A} \sum_{\forall S_k \in S} p_{a_i}(S_i, S_k)[R_{a_i}(S_i, S_k) + \gamma v_\pi(S_k)]$$

$$\pi(S_i) \leftarrow \operatorname*{argmax}_{a_i \in A} \sum_{\forall S_k \in S} p_{a_i}(S_i, S_k)[R_{a_i}(S_i, S_k) + \gamma v_\pi(S_k)]$$

end for

$\Delta \leftarrow max(\Delta, \parallel v_\pi - v \parallel_\infty)$

end while

4.4 Model-free reinforcement learning approach

In the case of model free reinforcement learning, the transmitter operates in an unknown environment trying to maximize the cumulative long term discounted reward by performing actions according to the observation of current states and rewards [135] [136].

In this section, we focus on the case that our system is model free and formulated as a discounted finite Markov decision process (MDP) and we adopt the reinforcement learning field to resolve the optimization problem being treated.

We have used Q-Sarsa algorithm in the first contribution [137] and Sarsa algorithm in the second contribution [138]. Both algorithms are well known methods for solving MDP problems in the context of model free conditions. Sarsa is an on-policy algorithm and Q-Sarsa mixes both Q-learning and Sarsa learning approaches in one algorithm. In fact, Q-Sarsa and Sarsa convergence to the optimal action-state values is guaranteed if some conditions are verified such as [139]: 1) S and A are finite, 2) The reward function is bounded 3)The policy is greedy in the limit with infinite exploration, 4)And the Robbins-Monro sequence of learning rate factor is verified as given in Equations 4.10.

$$\begin{aligned} &\sum_{n} \alpha_i(S_i, a_i) = \infty, \\ &\sum_{n} (\alpha_i(S_i, a_i))^2 < \infty \\ &w.p.1 \\ &\forall (s, a) \neq (S_i, a_i), \alpha_i(s, a) = 0 \end{aligned} \tag{4.10}$$

To be able to apply Sarsa and Q-Sarsa algorithms, we first assume that our transmitter is able to observe the components of the state S_n and the reward $R(S_n, S_{n+1})$ after having performing an action a_n in T_{Sn}. The reward is represented by the transmitted data packet size d_n which is supposed already known at the transmitter.

Equation 4.6 shows that $q(S_n, a_n)$ relative to the current action-state pair can be represented in terms of the expected reward of the current action-state pair and the state value function $v(S_{n+1})$ of the next state which is also equal to $v(S_{n+1}) = q(S_{n+1}, a_{n+1})$. Sarsa's

algorithm policy has an on-policy aspect that comes from the fact that the following Q-value $q(S_{n+1}, a_{n+1})$, which encompasses all long-term future rewards, is the result of the adoption of a_{n+1} actions derived from the π policy which is being followed at the same time. Thus, to make the optimal decision one should choose the action that maximize the reward and than continue following the optimal policy.

By using the $\epsilon - greedy$ policy, the decision made is conditioned by a an expansion of the set of actions being considered by balancing the exploration and exploitation approaches as long as the episodes are going on. Finally, the optimal policy can easily be derived from the $q^*(S_i, a_i)$ being obtained.

Based on this method, Sarsa and Q-Sarsa are acting in an iterative manner to estimate the optimal $q^*(S_i, a_i)$ over the whole set of available states being visited that form the learning sequences. In the n^{th} learning episode $q_n(S_n, a_n)$ is estimated by his current value and updated by the resulting error between the expected estimation of the next Q-value ($q(S_{n+1}, a_{n+1})$ added to the current reward and his previous estimate $q_{n-1}*(S_n, a_n)$. In each T_S, the resulting steps are [140]:

- observe the current state $S_n \in S$,
- choose and perform an action $a_n \in A$,
- observe the next state $S_{n+1} \in S$ and the immediate reward $R_{a_n}(S_n, S_{n+1})$
- updates the estimate of $q(S_n, a_n)$ using Sarsa approach:

$$q(S_n, a_n) \leftarrow q(S_n, a_n) + \alpha[R_{n+1} + \gamma q(S_{n+1}, a_{n+1}) - q(S_n, a_n)] \tag{4.11}$$

- updates the estimate of $q(S_n, a_n)$ using Q-Sarsa approach:

$$q(S_n, a_n) \leftarrow q(S_n, a_n) + \alpha[R_n + \gamma(sigma * q(S_{n+1}, a_{n+1}) + (1 - sigma) \times \max_{a_i \in A} q(S_{n+1}, a_i) - q(S_n, a_n)] \tag{4.12}$$

where α is the learning rate factor. If all actions are selected and performed with a non-zero probability (i.e. $0 \leq \gamma \leq 1$) and the conditions of convergence respected, than the sequence of $q_i(S_i, a_i)$ is proven to converge to $q^*(S_i, a_i)$ with probability 1 as $i \to \infty$ according to the Robins-Monro sequence given above.

Both Sarsa and Q-Sarsa are detailed in Algorithms 9 and 10 respectively in case of a learning sequence of 200.000 iterations length (e.g. 2000 episodes of 100 transitions).

Algorithm 9 Sarsa algorithm

1. Initialization stage:

for each state $S_i \in S$ and each action $a_j \in A$ **do**

Initialize $q(S_i, a_j)$ arbitrarily from $\mathbb{R}$ such that $q(S_{tr}, .) = 0$ for all terminal states S_{tr}

end for

2. Sarsa optimization stage:

$S_i \leftarrow$ some start state

$N_L \leftarrow 200.000$ as the maximum number of considered learning iteration

$l = 0$

$a_j \leftarrow \pi(S_i)$, where π is an ϵ-greedy policy based on Q

while $l < N_L$ **do**

$l = l + 1$

Take action a_j and observe reward R_l and next state S_k

Choose an action $a_j^{'} = \pi(S_k)$, where π is an ϵ-greedy policy based on Q

$q_{l+1}(S_i, a_j) \leftarrow q_l(S_i, a_j) + \alpha[R_l + q_l(S_k, a_j^{'}) - q_l(S_i, a_j)]$

$S_i \leftarrow S_k$

$a_j \leftarrow a_j^{'}$

end while

4.5 Numerical Results

We provide in this section an example of a communication system with energy harvesting as it was presented in section II. We are interested in comparing the performance achieved by the value iteration algorithm policy as a model based control and those of the Q-Sarsa

Algorithm 10 Q-Sarsa algorithm

1. Initialization stage:

for each state $S_i \in S$ and each action $a_j \in A$ **do**

Initialize $q(S_i, a_j)$ arbitrarily from $\mathbb{R}$ such that $q(S_{tr}, .) = 0$ for all terminal states S_{tr}

end for

2. Q-Sarsa optimization stage:

$S_i \leftarrow$ some start state

$N_L \leftarrow 200.000$ as the maximum number of learning iterations being considered

$l = 0$

$a_j \leftarrow \pi(S_i)$, where π is an ϵ-greedy policy based on Q

while $l < N_L$ **do**

$l = l + 1$

Take action a_j and observe reward R_l and next state S_k

Choose an action $a_j^{'} = \pi(S_k)$, where π is an ϵ-greedy policy based on Q

$q_{l+1}(S_i, a_j) \leftarrow q_l(S_i, a_j) + \alpha[R_l + (sigma * q_l(S_k, a_j^{'}) + (1 - sigma) * \max_{a_i \in A} q_l(S_k, a_i)) - q_l(S_i, a_j)]$

$S_i \leftarrow S_k$

$a_j \leftarrow a_j^{'}$

end while

and Sarsa algorithms being used in the case of a model free control.

We operate the calculation of the expected total transmitted data achieved in both cases by generating first a sequence of 2000 realizations of 100 arbitrarily state transition in order to average all the expected total transmitted data to guarantee a sufficient level of confidence on the obtained results.

The Q-Sarsa and Sarsa policies were generated for $\epsilon = 0.001$, 0.07 and 0.7, and then evaluated and averaged for 100 times to assess the associated performance when applied to the considered sequence stated before.

As a numerical values of different element of computation we have choose parameters based on IEEE 802.15.4e [141] for the time slot duration fixed at $\Delta T_S = 10ms$, the transmission period is of $\Delta Tx = 5ms$.

We assume that the packet sizes of data are in the set $D = \{300, 600\}$ and may vary according to the probability transition matrix $P_D = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$.

The fundamental energy unit of the harvester is assumed to be equal to $2.5\mu J$ according to the piezoelectric harvesting device in [118], and we suppose that the harvester being used either harvest 2 energy unit or does not harvest any one so the set of possible harvested energy amount is giving by $E = \{0, 2\}$ and the probability transition matrix is $P_E = \begin{bmatrix} pH_0 & 1 - pH_0 \\ 1 - pH_2 & pH_2 \end{bmatrix}$, where pH_0 represents the probability of harvesting 0 energy unit in the current time slot giving that no energy was harvested in the previous time slot as well as pH_2. The channel state is assumed to take two states as the set giving by $H = \{1.655 \cdot 10^{-13}, 3.311 \cdot 10^{-13}\}$ which may account for indoor channel model in urban scenarios cases in [143], the probability of channel state transition matrix is given by $P_H = \begin{bmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}$.

The required transmission energy elementary unit $E^{T_x} = e$ to successfully transmit a data packet of size 300 bits over a channel attenuation of $3.311 \cdot 10^{-13}$ in case of a noise power density of $10^{-20.4}$ (W/Hz) is of 2.5 μJ which may be equivalent to one unit harvesting energy. We assume that the transmission energy needed to guarantee a successful data packet transmission is an integer multiple of E^{T_x} and the energy spending over each

time slot may belong to the set $E^{T_S} = \{e, 2 \times e, 4 \times e\}$, which corresponds to a power consumption of 0.5, 1 and 2 mW respectively. The maximal capacity size of the battery is supposed to vary from 5 to 9 energy transmission units.

We set γ to 0.9 for all the algorithms being used. In particular, for the simulation in learning algorithm mode, we set the exploration appetite rate associated with the ϵ-greedy function to different values such as 0.7, 0.07 and 0.001 and the learning rate α is set to 0.5.

4.5.1 1st contribution numerical results

In this section the results of the publication titled "Policy iteration vs Q-Sarsa approach optimization for embedded system communications with energy harvesting" [137] are presented and discussed.

In Figure 4.2 we present the evolution of the expected cumulative transmitted data obtained using Q-Sarsa algorithm versus the number of the learning iteration elapsed time slots. The achieved results are compared to those of the optimal policy obtained using Policy Iteration algorithm. As we can see, Q-Sarsa needs first to run many learning episodes to explore the system behavior and elaborate successive policies that converge to an estimate of the optimal policy. With ϵ fixed to 0.7 high performance level is reached in a shorter learning time than with ϵ of 0.07 or 0.001, which was predictable as exploration actions are important for the optimal action findings and policy convergence. High ϵ generates decreasing performance at the steady regime due to the fact that the algorithm still trying non greedy actions. A progressive annealing of ϵ over time can resolve this problem.

By increasing the B_{max} characteristic of the battery, the expected result is improved, e.g. for epsilon of 0.07 or 0.001 as depicted in Figure 4.3. The Policy Iteration represents an upper bond performance to the Q-Sarsa algorithm.The cumulative transmitted data for Q-Sarsa with $\epsilon = 0.07$ and $N_L = 2000$ reaches almost 93 % of the Policy Iteration performance and only 78 % when ϵ is less important.

The high abundance of the harvested energy according to the pH_2 factor contributes to improve the results achieved by both PI and Q-Sarsa algorithms as depicted in Figure

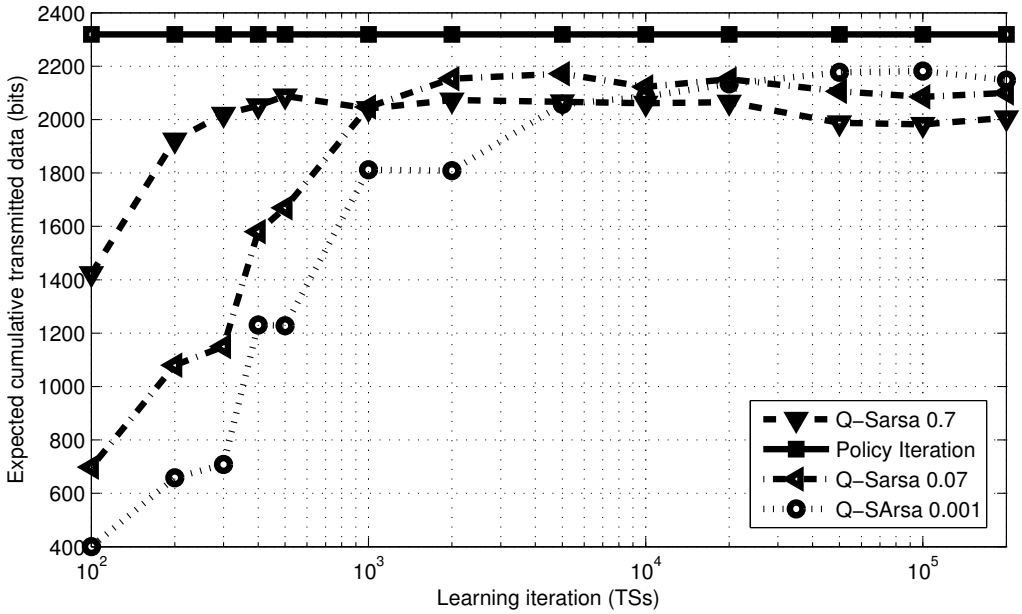

Figure 4.2: Expected Cumulative Transmitted Data versus the Learning Iteration time slot for $pH_0 = pH_2 = 0.9$ and maximal battery charge $B_{max} = 5$

4.4.

4.5.2 2nd contribution numerical results

In this section the results of the contribution in [138] are presented and discussed. In Figure 4.5 we represent the evolution of the performance obtained under Sarsa algorithm vs the learning iteration elapsed time (T_Ss). The achieved result is compared to that of the optimal policy calculated using Value iteration algorithm. As we can see Sarsa needs to run many episodes to learn the environment behavior before elaborating a policy that lead to maximize the obtained total cumulative reward. With ϵ fixed to 0.07 the rhythm of obtaining good result is faster than with ϵ fixed to 0.001 which illustrate the importance of exploration activities for approaching the optimal policy. High ϵ values generate oscillations at the study regime due to the fact that the algorithm still trying non greedy actions. A progressive annealing of ϵ over time can resolve this problem.

The VI approach take advantage of available information of the MDP process and represent an upper bond of the performance of those related to the Sarsa algorithm.The

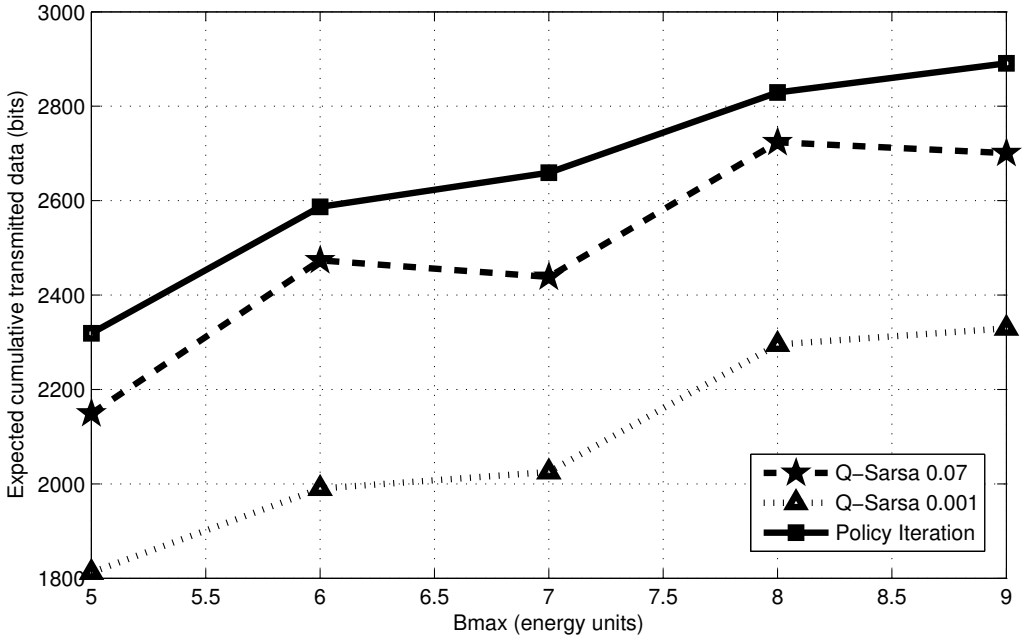

Figure 4.3: Expected Cumulative Transmitted Data versus maximal battery charge B_{max} where $pH_0 = pH_2 = 0.9$and $N_L = 2000$ iterations

resulting cumulative reward on Sarsa for $\epsilon = 0.07$ and $N_L = 1000$ reaches almost 89 % of the VI performance and only 78 % when ϵ is less important where N_L is total number of the learning iteration time slots (T_Ss) being conducted.

In average for $N_L = 1000$, the considered system operates by exploitation the major battery capacity charge available at the beginning to perform massive transmission and after that rely on the harvesting process to execute additive transmissions in order to improve the expected total transmitted data as depicted in Figure 4.6 and 4.7.

In Figure 4.8 we illustrate the evolution of the expected total transmitted data according to the B_{max} value, so we notice that the more the battery capacity is important the more the cumulative reward obtained is significant.

Giving that pH_2 gives the probability of a continuous energy harvesting process in each time slot. Its impact on the performance of the system is considerable to complete more transmission and to increase the expected cumulative reward. When pH_2 starts to increase, the impact is positive on the reward obtained by the system as shown in Figure 4.9.

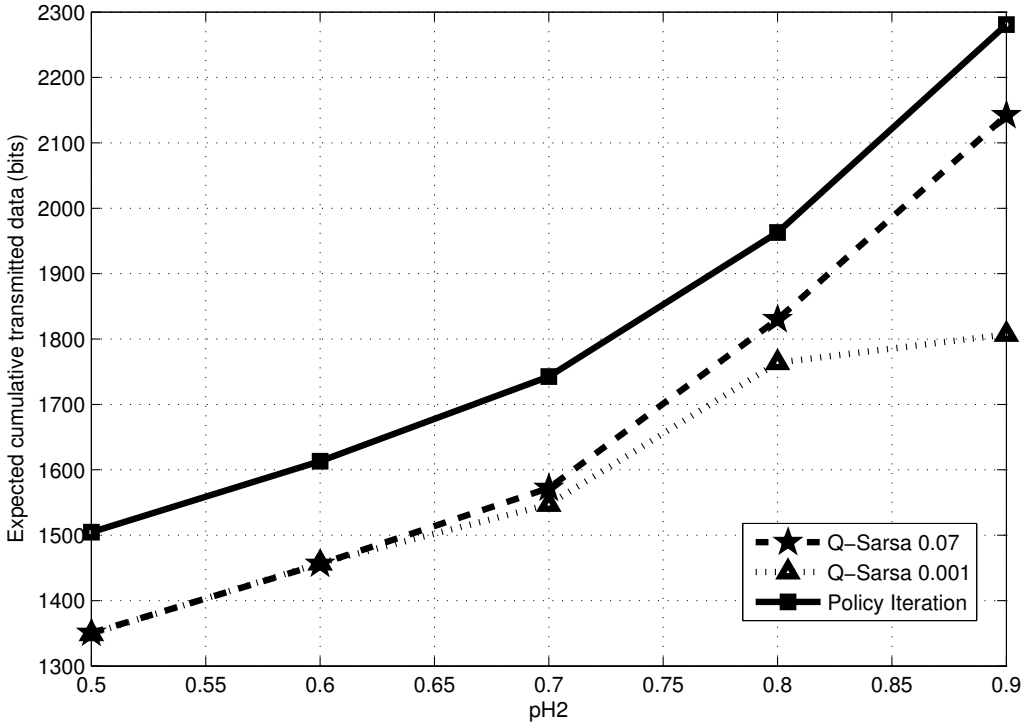

Figure 4.4: Expected Cumulative Transmitted Data versus channel transition probability pH_2 where $pH_0 = 0.9$, $N_L = 2000$ iterations and maximal battery charge $B_{max} = 5$

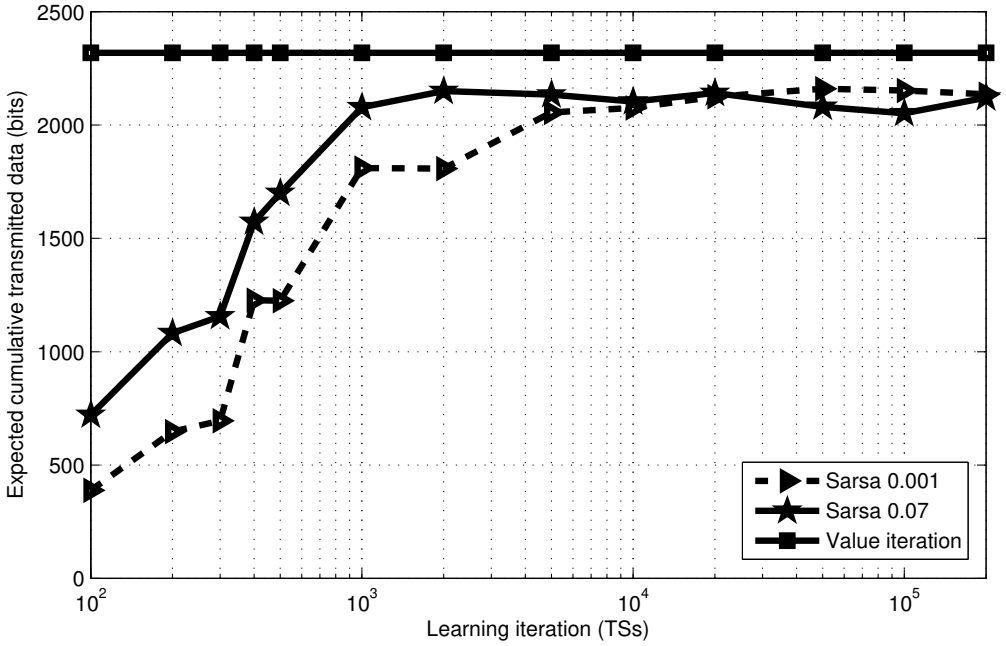

Figure 4.5: Expected total transmitted data vs Learning Iteration with $pH_0 = pH_2 = 0.9$ and $B_{max} = 5$

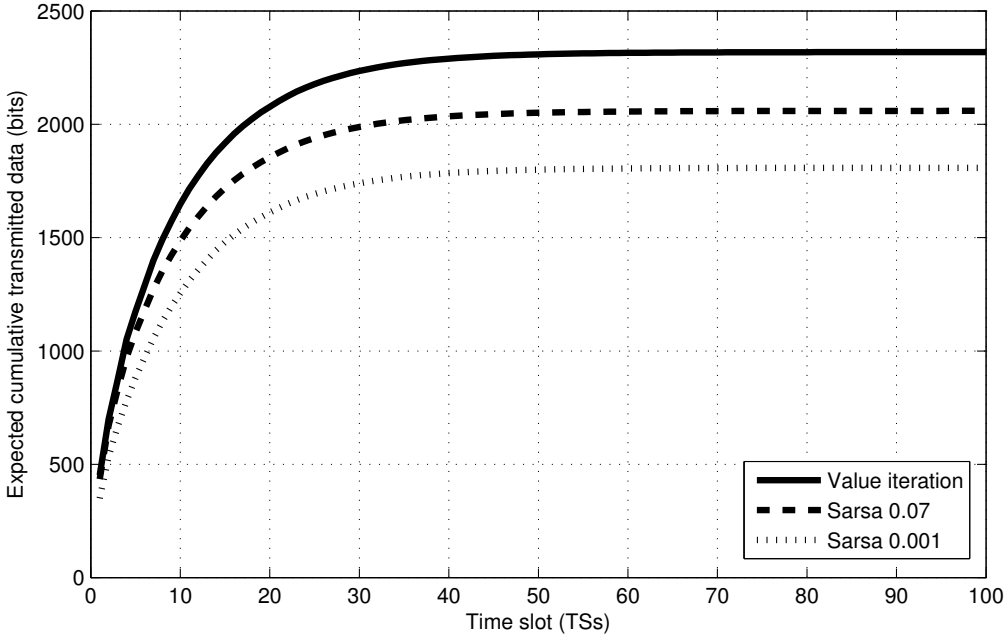

Figure 4.6: Expected total transmitted data vs T_Ss with $pH_0 = pH_2 = 0.9$ and $B_{max} = 5$

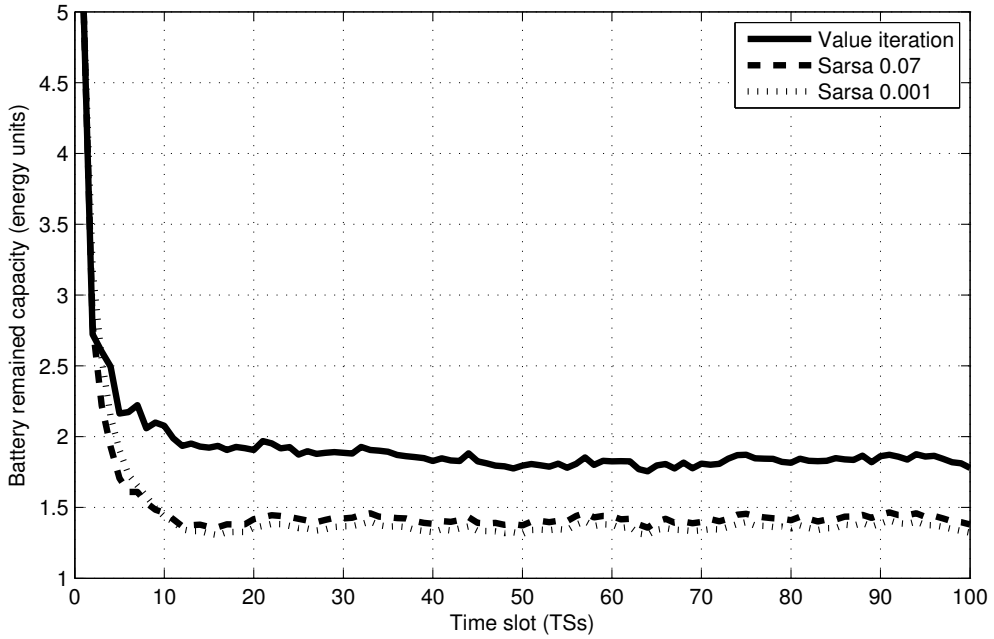

Figure 4.7: Battery consumption vs T_Ss with $pH_0 = pH_2 = 0.9$ and $B_{max} = 5$

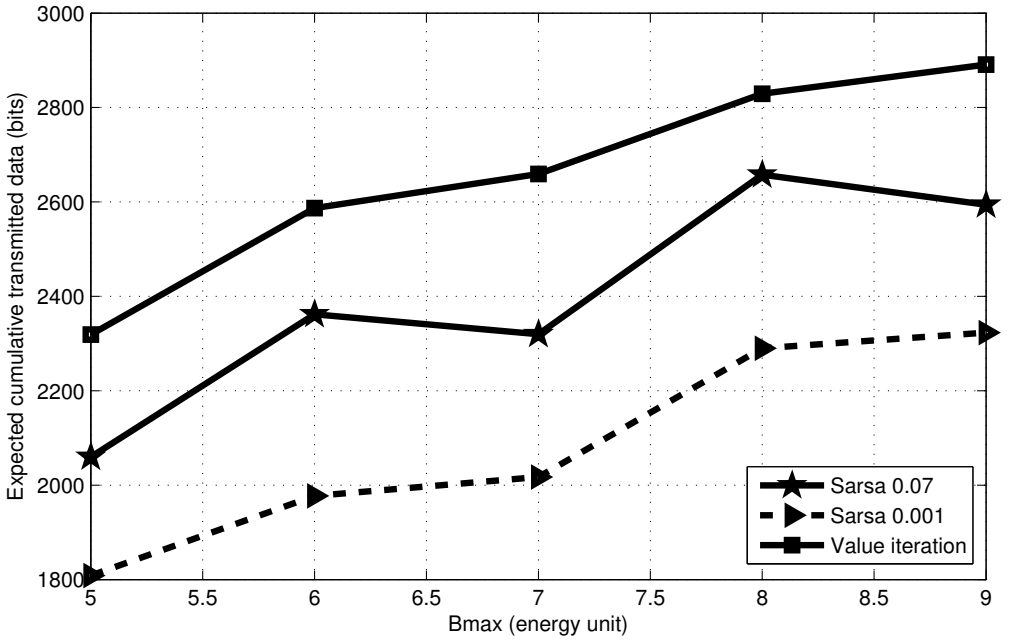

Figure 4.8: Expected total transmitted data vs B_{max} with $pH_0 = pH_2 = 0.9$ and $N_L = 1000$ iterations

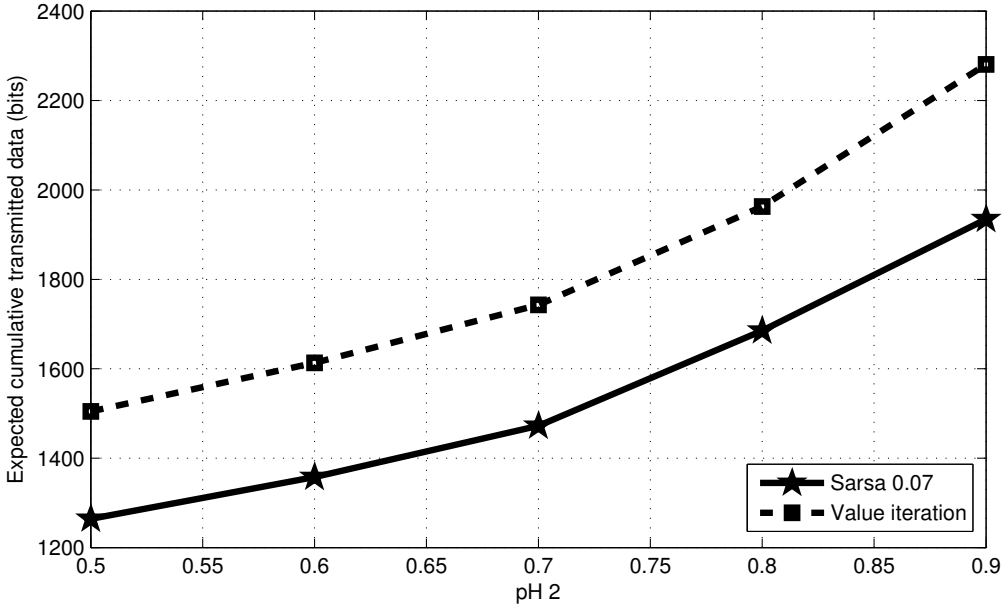

Figure 4.9: Expected total transmitted data vs pH_2 with $pH_0 = 0.9$, $N_L = 1000$ iterations and $B_{max} = 5$

4.6 Conclusion

In this chapter, we have addressed the problem of battery optimization with harvesting capacity in wireless sensor peer to peer communications by applying approaches from the fields of Dynamic Programming (DP), such as Policy iteration and value iteration, and from the field of Reinforcement Learning (RL) such as Q-Sarsa and Sarsa. In the first part of the simulation results, we applied the RL-based Q-Sarsa algorithm to a battery consumption optimization problem in the context of wireless sensor networks with energy harvesting. We compared the performances of the Q-Sarsa algorithm to those of the Policy Iteration algorithm. We concluded that even for model-free systems, the process of learning when performed by the transmitter converge to an optimal estimation of decision policies. Reasonable number of iterative learning time slots were sufficient to achieve good performances using Q-Sarsa algorithm. The obtained result converge to those of PI algorithm as the learning time is increased. We have proven that smart wireless sensors are themselves able to make good decisions when they are in front of such optimization problems. Based on experiences, the sensors have shown that they can learn how to

maximize there rewards by exploring the available space of action-state pairs and then exploiting the obtained results associated with the visited action-state pairs.

In the second contribution part, we have proven that through the implementation of practical algorithms in smart sensors for wireless systems, good performance can be achieved even in model-free cases where the sensor in question had no prior knowledge of the communication network and its environment by using a preliminary learning phase to fill this information gap. With the Sarsa algorithm, which comes from the field of reinforcement learning in machine learning techniques, a learning phase of only 10 episodes was sufficient to obtain performances close to those of the value iteration algorithm. The value iteration algorithm comes from the field of dynamic programming and is applied only when all the information is available in advance. The policies that were computed following an iterative reinforcement learning mechanism based on the experienced action-state pairs are an interesting alternative to the model based policy approaches.

In conclusion, the learning approaches seem to be adapted to real cases of operational environment, especially in the context of wireless sensor node communications, when the lack of knowledge is a fact, than the classical methods from dynamic programming field, e.g. policy iteration and value iteration algorithms.

Chapter

5 Embedded systems with recovery-aware Batteries

In this chapter, the recovery phenomenon is exploited to design an algorithm that optimizes both the lifetime of the battery and the performance of the studied system. The algorithms from Dynamic programming and Reinforcement learning fields are the first to be considered. When in Dynamic programming prior detailed information are assumed to be available, in reinforcement learning those information becomes unknown and long calculation times are needed to converge toward an optimal policy solution. A new Rapid Learning Algorithm (RLA) that combines both Dynamic programming and Reinforcement learning features is proposed. RLA exploits a reduced model of the system instead of exploring the whole and heavy system state model as Dynamic programming do. The RLA run-time is then considerably shortened.

5.1 Introduction

In this chapter, we aim to maximize the obtained rewards when only one battery is used. We are interested in exploiting the recovery effect that occurs inside the batteries to extend lifetime durations and enhance the achieved rewards. The system that support our study is given in Fig. 5.1 where d_n represents the received data packet sizes, h_n the channel state and a_n the transmission action being taken. The system is operating in a time slotted way with equal duration T_S. Three cases are then considered depending on

the information availability and described as follows:

- The first case is a fully known environment case study, in which we assume that the optimizer has prior stochastic knowledge of data packet size and channel state transitions. Value Iteration from Dynamic programming field is appropriated for this kind of situation ;
- The second case is a fully unknown environment case study problem. The sensor node is assumed to have no prior information about the system state transitions. Therefore, the optimizer conducts first a learning episodes to model the system states behavior before calculating the optimal strategy to implement using reinforcement learning algorithms such as Q-learning or Sarsa ;
- In the context of RLA, which is the third case study, the optimizer exploits partial system states information being available and learn only the unknown parts left.

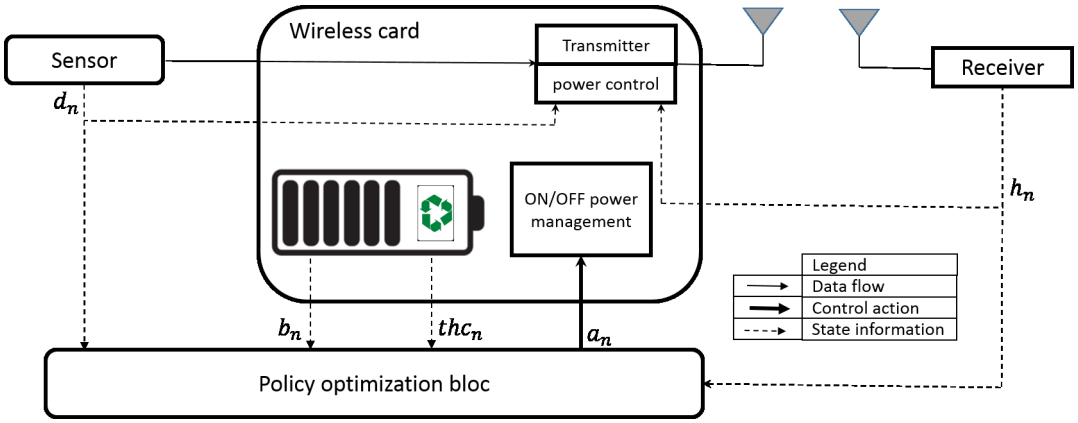

Figure 5.1: The recovery-aware battery powered study system

In the remainder parts of the chapter, Section 5.2 describes in details the whole system model that support our study. Dynamic Programming and Reinforcement Learning approaches are presented in Section 5.3 and Section 5.4 respectively. The obtained results from simulations are then presented and discussed in Section 5.5. In Section 5.6, the concept of partially known model is detailed and its numerical results are presented in Section 5.7. And finally, we conclude the chapter in Section 5.8.

5.2 System model description

In this chapter, we assume that the battery has limited storage capacity and will not be subject of replenishment by any external charger. Our main problem is to optimize the use of all the quantity of charge units available inside the battery simultaneously with the maximization of the amount of cumulative transmitted data during the whole lifetime of the system.

As for the previous chapter, we suppose that the data arrives in packets at each time slot T_S with different sizes that we model as a stochastic process characterized by a first order Markov model. A received data packet is supposed to be transmitted within the next T_S duration otherwise it will be destroyed.

We denote by e, the elementary battery charge unit that is sufficient to perform one data packet successful transmission with a basic size. e is also assumed to be the quantity of charge being harvested during each idle time-slot T_S.

A data packet transmission is declared to be successful when a sufficient charge unit quantity, available in the battery, is consumed. The quantity of energy needed depends on the received data packet size and the channel state. It is obvious that the more data packet size is important the more the charge units quantity needed for successful transmission is large. So is the case for the channel states with higher attenuation power. The battery charge levels information as well as the data packed sizes and the channel states are considered to be available at the transmitter at each time slot. The information about the channel states is supposed to be shared between the transmitter and the receiver. For decision making, the transmitter exploits the information about the system state components such as data packet size available for transmission, the wireless channel state and the quantity of energy units still available in the battery. The transmitter choose either to transmit the data packet or to drop it following the implemented policy. This policy guides the transmitter to make optimal decisions that allow it to avoid bringing the system into trapping states and wasting the battery capacity. We denote by a_j the action being taken by the transmitter, so when transmission action is taken (a_j is set to 1, otherwise ($a_j = 0$). (a_j can have only two different values that form the set $A = \{0, 1\}$.

The quantity of charge units needed for transmission as well as the battery charge state constraints are subject to the equations giving in 5.1, 5.2 and 5.3.

$$a_n E_n^{T_S} \leq B_n, \tag{5.1}$$

$$B_n \leq B_{max}, \tag{5.2}$$

$$\sum_{n=1}^{N_L} a_n E_n^{T_S} \leq T_B. \tag{5.3}$$

With B_n the amount of charge units available for use in the battery and N_L the total number of performed iteration before the battery becomes completely exhausted. Battery charge state B_n and consumed theoretical capacity thc_n are supposed to be known at the beginning of each time slot. They change only after performing a transmission or a charge recovery when trapping state has not been reached yet. The battery charge state are in the set $B = \{0, 1, 2, 3, ..., B_{max}\}$.

The battery charge units amount B_n and the consumed theoretical capacity thc_n are updated after each iteration according to Equations 5.4 and 5.5 respectively.

$$B_{n+1} = \begin{cases} B_n - E_n^{T_S} & \text{if } a_n = 1 \\ min\left(B_n + e, B_{max}\right) & \text{if } a_n = 0 \end{cases} \tag{5.4}$$

$$thc_{n+1} = \begin{cases} max\left(thc_n + E_n^{T_S}, T_B\right) & \text{if } a_n = 1 \\ thc_n & \text{if } a_n = 0 \end{cases} \tag{5.5}$$

The optimization goal for the cumulative transmitted data packets over the whole lifetime of the battery and hence the entire embedded system is giving in Equation 5.6.

$$\max_{\{a_i\}_{i=1}^{N_L}} \sum_{n=1}^{N_L} \gamma^{n-1} a_n \times d_n \tag{5.6}$$

$$\text{subject to Equations 5.1, 5.2, 5.3, 5.4 and 5.5} \tag{5.7}$$

Where $0 \leq \gamma \leq 1$ is the discounted factor over time for the total system rewards.

The considered system operates according to a discrete-time way and has a set of states S_n with finite size that can be modeled as Markovian Decision Problem (MDP). When provided, the MDP constitutes the background platform for solving the optimization

problem. The system is also considered fully observable so that the next system state and reward are known by the transmitter as decision maker agent. The system states transition probability as they will be defined later are also modeled as a first order Markovian process. For a time slot T_{S_n}, we denote by S_n the correspondent state of the system. We assume that S_n encompasses different element as battery charge level, consumed theoretical capacity, channel state and data packet size, so that $S_n = (B_n, thc_n, h_n, d_n)$. We denote by S the set of all the system states so that $S = \{S_1, S_2, ..., S_{N_S}\}$ where N_S is the total number of all possible system states.

The obtained MDP model can be described by the quintuple $\langle S, A, P, R, \gamma \rangle$, where:

- S represents the set of system states with finite size,
- A represents the set of actions being available with finite size,
- **P** represents the state transition probability matrix which is defined by $p_a(s, s') \triangleq \mathbf{P}(S_{j+1} = s' | S_j = s, a_j = a)$, where a $\in$ A and $(s,\ s') \in$ S$\times$S.
- R is the reward function, such that $R_a(s, s') \triangleq \mathbf{E}(R_{j+1} | S_j = s, a_j = a)$
- γ is a discount factor with $\gamma \in [0, 1]$

We denote by policy π the rule function that matches each system state S_n to the decision that should be made by the transmitter. The obtained reward is a function that evaluates the current action-state pair. In our case, the obtained reward is represented by a single number which is the quantity of bits having been transmitted when evolving from one system state to another according to the decision made by the transmitter. This reward function is denoted by $R_n = a_n \times d_n$. The total transmitted data is the sum of all the obtained rewards mitigated by mean of the discounted factor γ. The transmitter policy value π is given by using the state-value function denoted by $v_\pi(S_i)$. This latter represents the expected cumulative rewards when we start from state S_i and we choose actions according to the policy π. This function $v_\pi(S_i)$ is defined in (5.8) and will be our main tool for obtaining the optimal policy π^* that solves the optimization problem.

$$v_\pi(S_i) \triangleq \sum_{\forall S_k \in S} p_{\pi_{(S_j)}}(S_i, S_k)[R_{\pi_{(S_j)}}(S_i, S_k) + \gamma v_\pi(S_k)] \tag{5.8}$$

We use also the action-state value function to evaluate the obtained cumulative rewards when we start from a state S_i and take an action a_i and after that we choose actions following the policy π. This action-state value function is giving in (5.9).

$$q^{\pi}(S_i, a_j) \triangleq \sum_{\forall S_k \in S} p_{a_j}(S_i, S_k)[R_{a_j}(S_i, S_k) + \gamma v_{\pi}(S_k)] \tag{5.9}$$

A policy π' is said to be better than or equal to a policy π, denoted by $\pi' \geq \pi$, if and only if the expected cumulative rewards obtained following the policy π' is greater than or equal to that obtained with policy π which is equivalent to have $v_{\pi'}(S_i) \geq v_{\pi}(S_i)$. The optimal policy π^* is defined as the policy which is better or equal to any other possible policy. The corresponding optimal state value functions $v_{\pi^*}(S_i)$ with i=1, 2, ...,N_S is also the better. In addition, this latter can be deduced from the action-state value function as given in (5.10).

$$v_{\pi^*}(S_i) = \max_{a_i \in A} v^{\pi^*}(S_i, a_i) \tag{5.10}$$

According to a system state S_i, the optimal policy is the greedy function over the set of action-state pair values of all possible combinations (S_i, a_j). At each step the transmitter should answer the question about which action can maximize its immediate reward and thereafter continues its trajectory following the policy π. Hence, to reach to optimal expected cumulative rewards, the transmitter should always choose the action obtained according to the greedy action and then follow the optimal policy π^* as giving in 5.11.

$$\begin{aligned} q^{\pi^*}(S_i, a_j) = \sum_{\forall S_k \in S} p_{a_j}(S_i, S_k)[R_{a_j}(S_i, S_k) + \\ \gamma \max_{a_k \in A} q^{\pi^*}(S_i, a_k)] \end{aligned} \tag{5.11}$$

The optimal policy is then obtained as given in Equation 5.12.

$$\pi^*(S_i) = \underset{a_j \in A}{\text{argmax}} q^{\pi^*}(S_i, a_j) \tag{5.12}$$

We will first discuss the first and second scenarios in sections 3 and 5. The third scenario will be introduced and detailed up to section 6. As expected, in the first scenario, we assume that we have prior knowledge of the stochastic behavior of the system with available information provided about the different components of the $\langle S, A, P, R, \gamma \rangle$ MDP

model . Thus, we will be able to apply the Dynamic programming algorithms in order to find the optimal transmission policy π^*. In the second scenario, we assume that no such prior information on the system behavior is available. We will then manage to explore and exploit the set of information resulting from transmission experiences. Following a Reinforcement learning approach, the obtained policy to be followed will be an estimation of the optimal policy π^*.

5.3 Fully known environment case study: Model-based algorithm

Relying on the stochastic model, we can track the battery discharge evolution and the recovery effect impact. As stated earlier, the battery is modeled according to the stochastic model as a discrete Markov chain process. We notice in Fig. 3.11, that $(B_{max}+1)$ Markov states are considered and numbered from 0 to B_{max} which represents the usable charge inside the battery. The charge unit number B_{max} represents the nominal capacity of the battery which means that B_{max} charge units can be drained from the battery with constant current load and a full charge starting point. The full charged battery state is denoted by state B_{max} and the exhausted battery state by 0. We consider that transmitting one data packet over the channel consumes k charge units and causes the transition of the battery from the current state i towards state $i-k$, as shown in Fig. 5.2 for $k=2$ and in Fig. 5.3 for $k=4$.

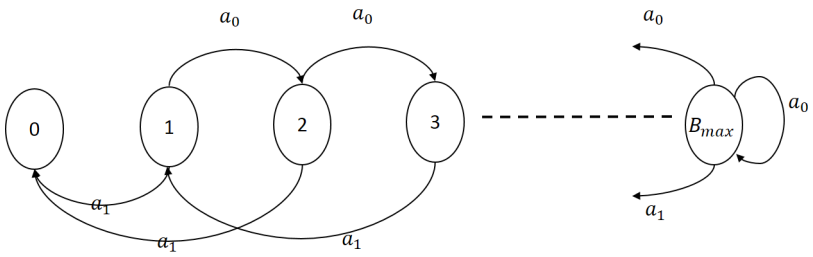

Figure 5.2: Two units discharge (k=2) and one unit recovery model within a T_S

We use algorithms derived from the Dynamic Programming to solve the MDP problem

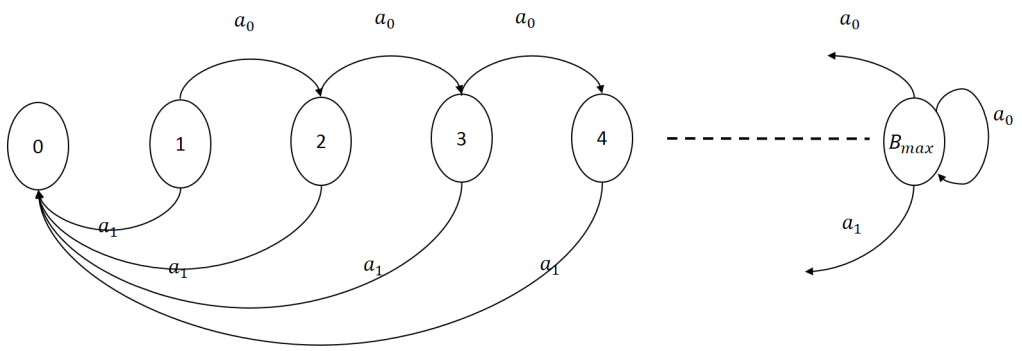

Figure 5.3: Four units discharge (k=4) and one unit recovery model within a T_S

resulting from Equation 5.6. Value Iteration algorithm (VI),originated from DP, is considered in this chapter. Our MDP system satisfies all conditions of optimization convergence. It has finite action and state spaces, a stationary and bounded reward function and a discount factor $0 \leq \gamma \leq 1$. Hence, it is proven that VI converge to the optimal policy [64]. The solution considered at this stage is to combine the set of equations composed of 5.1, 5.2, 5.3, 5.4, 5.5, 5.10 and 5.11 to obtain the optimal policy. The key idea of DP algorithms such as VI algorithms is to use the value functions to structure and organize the search of the optimal policy. By following this latter, the transmitter will be able to maximize the cumulative obtained rewards and then will meet the optimization objective.

To use Value iteration algorithm, we will exploit Equation 5.10 and the evaluation process of $v_{\pi_l}(S_i)$ will be conducted as in (5.13).

$$\begin{aligned} v_{\pi_l}(S_i) &= max_{a_i \in A} q_l^{\pi}(S_i, a_i) \\ &= max_{a_i \in A} \sum_{\forall S_k \in S} p_{a_l}(S_i, S_k)[R_{a_l}(S_i, S_k) + \\ &\gamma max_{a_i \in A} q_{l-1}^{\pi}(S_i, a_i)] \end{aligned} \tag{5.13}$$

For all $S_i \in S$, the sequence $\left\langle v_{\pi_0}, v_{\pi_1'}, v_{\pi_2''}, ..., v_{\pi_k^{(k)}}, ... \right\rangle$ will converge to v_{π^*} as well as the conditions of convergence are met. In this chapter, The algorithm that has been implemented in the system decision making is Value iteration as detailed in Algorithm 8.

5.4 Fully unknown environment case study: Model-free algorithm

Now, the transmitter is wandering in an unknown environment. It will experiment actions and observe resulting rewards to learn the system model before trying to find a policy that maximizes the total cumulative transmitted data. In this section, we adopt the reinforcement learning approaches to solve the optimization problem in hands . We use both Q-learning and Sarsa algorithms which are two of the most used methods for solving MDP problems in model-free context. Sarsa is called after the succession of states, actions and rewards when following the policy π being generated and is acting as an on-policy strategy. Q-learning is similar to Sarsa but acts following an off-policy strategy.

The learning algorithms are based on temporal estimation of the state value function $v_\pi(S_i)$ that provides the expected cumulative rewards when the agent starts in state S_i and follows policy π thereafter. Also, the action state value $q_\pi(S_i, a_i)$ provides an evaluation of the expected cumulative rewards when the agent take first an action a_i when being at state S_i and then follows policy π. This two value functions are related to each other through Equation 5.14.

$$v_\pi(S) = \sum_{a_i} Prob(a_i/S_i)\, q_\pi(S_i, a_i) \tag{5.14}$$

The learning method converges to the optimal action value function $q^*(S_i, a_i)$ from which an optimal policy π^* can easily be deduced. $q^*(S_i, a_i)$ is the convergence iteration result of $q_l(S_i, a_i)$ updating process. For Sarsa in Algorithm 11, the updating formula of the action-state value function is detailed in 5.15.

$$\begin{aligned} q(S_n, a_n) \leftarrow q(S_n, a_n) + \alpha[R_{n+1} + \\ \gamma q(S_{n+1}, a_{n+1}) - q(S_n, a_n)] \end{aligned} \tag{5.15}$$

Where α is a learning rate factor.

For Q-learning in Algorithm 12, the updating action-state value function formula is de-

tailed in 5.16.

$$q(S_i, a_i) \leftarrow q(S_i, a_i) + \alpha[R_{i+1} + \gamma \max_{a_j} q(S_{i+1}, a_j) - q(S_i, a_i)] \tag{5.16}$$

Algorithm 11 Sarsa algorithm

1. Initialization stage:

for each state $S_i \in S$ and each action $a_j \in A$ **do**

Initialize $q(S_i, a_j)$ arbitrarily from $\Re$ such that $q(S_{terminal}, .) = 0$ for all terminal states $S_{terminal}$

end for

initial time index setting $l \leftarrow 0$

2. Sarsa episodic learning step:

$S_i \leftarrow$ some start state

$a_j \leftarrow \pi(S_i)$, where π is an ϵ-greedy policy based on Q

while *S_i is not a terminal state & the Battery theoretical capacity is not reached yet & the episode number* $< N_L$ **do**

Take action a_j and observe reward R_l and next state S_k

Choose an action $a'_j = \pi(S_k)$, where π is an ϵ-greedy policy based on Q

$q_{l+1}(S_i, a_j) \leftarrow q_l(S_i, a_j) + \alpha[R_l + q_l(S_k, a'_j) - q_l(S_i, a_j)]$

$S_i \leftarrow S_k$

$a_j \leftarrow a'_j$

Update the time index $l \leftarrow l + 1$

end while

5.5 Numerical Results: 3rd contribution

In this section the results of the publication titled "Battery Recovery-Aware Optimization for Embedded System Communications" [142] are presented and discussed.

We will provide first all the features of our considered point to point system communication model and then we compare the performances achieved by the Value iteration, greedy, Sarsa and Q-learning algorithms. Actually, for comparison purposes, we will consider a

Algorithm 12 Q learning algorithm

1. Initialization step:

for each state $S_i \in S$ and each action $a_j \in A$ **do**

Initialize $q(S_i, a_j)$ arbitrarily from $\Re$ such that $q(S_{terminal}, .) = 0$ for all terminal states $S_{terminal}$

end for

initial time index setting $l \leftarrow 0$

2. Q learning optimization step for each episode:

$S_i \leftarrow$ some starting state

while *S_i is not a trapping state & the Battery theoretical capacity is not reached yet & the episode number* $< N_L$ **do**

$a_j \leftarrow \pi(S_i)$, where π is an ϵ-greedy policy based on Q

Take action a_j and observe reward R_l and next state S_k

$q_{l+1}(S_i, a_j) \leftarrow q_l(S_i, I_j) + \alpha[R_l + q_l(S_k, a'_j) - q_l(S_i, I_j)]$

$S_i \leftarrow S_k$

Update the time index $l \leftarrow l + 1$

end while

greedy algorithm where the transmitter is always making the decision of transmitting any received data packet whatever the battery available storage, the packet size and the channel state are $(a_i = 1, i = 1 : \infty)$.

The results of the expected cumulative transmitted data, consumed capacity in terms of charge units, sensor lifetime before the battery becomes completely discharged are all obtained from the simulation. In all cases, 10000 episode realizations, with 100 transitions each, is generated and used for simulation purposes. This sequences are produced according to the probability transition matrices of both data packet sizes and channel states.

VI optimal policy, greedy, Q-learning and Sarsa algorithms would be running first with parameters $B_{max} = 5$, $\epsilon = 0.001$ and 0.7. A sequence of $N_L = 10000$ episodes will be considered to achieve sufficient knowledge of the system as given in Figure 5.8 and 5.9. Then we will vary the battery nominal capacity B_{max} from 5 to 9 with $\epsilon = 0.7$ for Optimal and greedy algorithms. For Q-learning an Sarsa algorithms, we set $N_L = 1000$ when B_{max} is varying from 5 to 9 charge units.The resulting outcomes are computed up to 100 times before being averaged.

As additional numerical values of different elements of computation we have choose parameters that are based on IEEE802.15.4e [141] for the time slot duration which will be fixed at $\Delta T_S = 10$ ms, the transmission period will be set to $\Delta Tx = 5$ ms. We assume that the data packet sizes are in the set $D = \{300, 600\}$ with only two possible realizations. They may vary according to a probability transition matrix that is equal to $P_D = \begin{bmatrix} v_1 & v_2 \end{bmatrix}$, where $v_1^t = \begin{bmatrix} 0.9 & 0.1 \end{bmatrix}$ and $v_2^t = \begin{bmatrix} 0.1 & 0.9 \end{bmatrix}$. The channel state is assumed to be in two possible states too and we set $H = \{1.655 \cdot 10^{-13}, 3.311 \cdot 10^{-13}\}$ which may account for an indoor channel model in urban scenarios cases like in [143], the transition state probability of the channel is given by the matrix $P_E = \begin{bmatrix} v_1 & v_2 \end{bmatrix}$. The minimum energy e that is required for a successful transmission of a data packet of size $d_n = 300$ bits over a channel with state $h_n = 3.311 \cdot 10^{-13}$ with a noise power density set to $N_0 = 10^{-20.4}(W/Hz)$ will be equal to $\frac{d_n log(2) N_0}{h_n} = 2.5\mu J$. The energy needed for successful transmission depends on the data packet sizes and channel states. Hence, it is a multiple integer of the energy unit e that belongs to the set $\{1, 2, 4\}$. This quantities

correspond to an energy consumption of $2.5, 5$ and $10\mu J$ and are equivalent to power of $0.5, 1$ and $2mW$ respectively.

We set γ at 0.9, as we consider a discounted communication process over time, for all the algorithms being used. Also, for learning algorithms, we set $\epsilon - greedy$ rate first at 0.7 and then 0.001 and α learning rate at 0.5.

In Fig. 5.4, for a theoretical capacity value of 50 charge units, Q-learning and Sarsa algorithms for low $\epsilon - greedy$ values, i.e. $\epsilon = 0.001$, produce closer results as learning episodes number N_L is varying from 1 to 10000 episodes. The optimal Value Iteration algorithm generates a policy that allows the sensor transmitter to send received data packets while being aware to avoid the battery trapping state 0 as long as the whole theoretical capacity is not totally consumed and performs far better that the greedy algorithm with only 47% of the optimal performance. For high $\epsilon - greedy$ rate value of the learning process, such as 0.7 which allows more state explorations, the design of the convenient policy is more indicated with Q-learning algorithm. This latter provides better cumulative reward levels and has even reached 95% of the performance achieved by the optimal Value Iteration approach where Sarsa reach only 79% of this performance for $N_L = 1000$ learning episodes.

For low $\epsilon - greedy$ rate are used, Sarsa and Q learning algorithms are then both indicated to generate the transmission policy in such case where a limited exploratory appetite is allowed. Those algorithms provide good cumulative reward levels and have even reached 93% of the performance achieved by the optimal Value Iteration approach for $N_L = 1000$ learning episodes.

In Fig. 5.5, the energy units consumption for optimal policy represents an upper bound consumption level for all other algorithms. The previous observations are confirmed by the level of the usable battery charge units exploited by Q-learning, after $N_L = 1000$ learning episodes, of 100% of theoretical capacity with $\epsilon = 0.7$ similar to the performance achieved by the Value Iteration algorithm. The greedy algorithm is the worst solution to consider since its performance is far away behind those of all other algorithms with a usable charges equal to 10% of the theoretical capacity.

Fig. 5.6 illustrates the evolution of the obtained cumulative rewards for all the algorithms

being considered when B_{max} changes from 5 to 9 charge units. The results obtained shows a smooth growth of the amounts achieved by the Value iteration cumulative rewards, as well as for Sarsa and Q-learning algorithms. This is mainly justified by an increased number of transmission opportunities offered by the policies for high B_{max} values with the same value of the theoretical battery capacity of 50. The greedy algorithm still achieve bad results when B_{max} varies from 5 to 9 charge units far away behind all other algorithms.

In Fig. 5.7, we see high performance obtained by both Optimal policy and Q learning algorithm with $\epsilon = 0.7$ as all the theoretical capacity has been exploited. Sarsa algorithm is achieved a percentage of used capacity of 90% when $B_{max} = 5$ and 94% when $B_{max} = 9$ for $N_L = 1000$ learning episodes.

In Fig. 5.8, we notice a saturation of the amount of transmitted data as the theoretical capacity exceeds 40 charge units. This can be explained by the constraint that data is worthy as long as it is transmitted earlier in time according to the discount factor γ. Optimal policy still the upper-bound of the obtained results then Q-learning policy and Sarsa. The choice of the battery should consider this saturation to optimize the cost of investment as it is worthless to use batteries with high theoretical capacities. The greedy policy brought the worst results as it was obtained before.

Fig. 5.9 shows that the optimal and Q-learning policy are similar in terms of used battery's capacity when T varies from 10 to 50 charge units. Sarsa doesn't do as well as Q-learning for $\epsilon = 0.7$. The greedy policy gives a constant amount of transmitted data even if T is growing.

Long battery lifetimes are obtained when optimal policy, Q-learning or Sarsa are considered as depicted in Fig.5.10. For battery with $B_{max} = 5$, it lasts for 744 ms, 759 ms and 720 ms for optimal policy, Q-learning and Sarsa respectively. The greedy algorithm still coming far behind and shows the shortest battery lifetime with 308 ms in similar conditions.

In Fig.5.11, an upper-bound for the obtained throughput is given by the optimal policy then Q-learning and Sarsa. For $B_{max} = 5$, optimal policy offers an average throughput of 1.857 Kbits/s, Q-learning offers 1.72 Kbits/s and Sarsa 1.536 Kbits/s. The greedy policy offers only 0.885 Kbits/s as throughput.

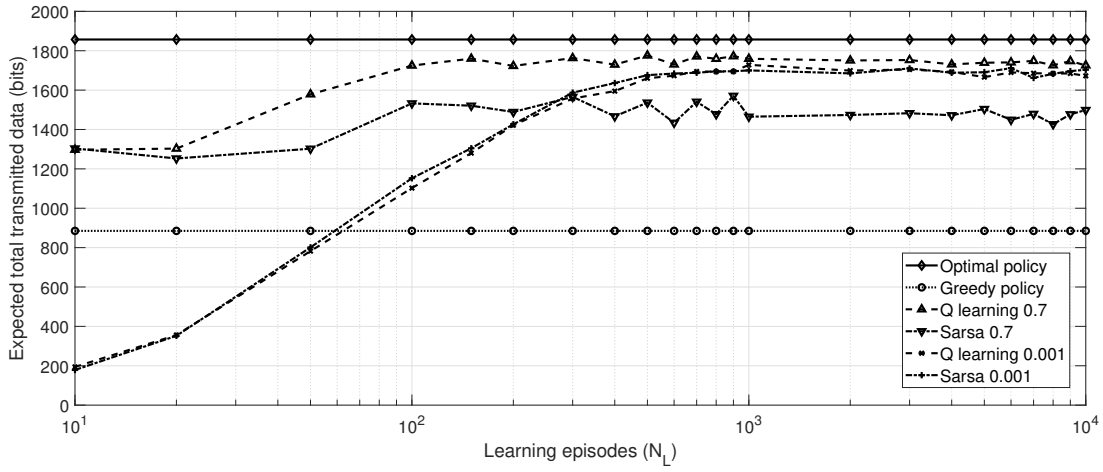

Figure 5.4: Expected total transmitted data vs learning episodes with $B_{max} = 5$ and $T_B = 50$

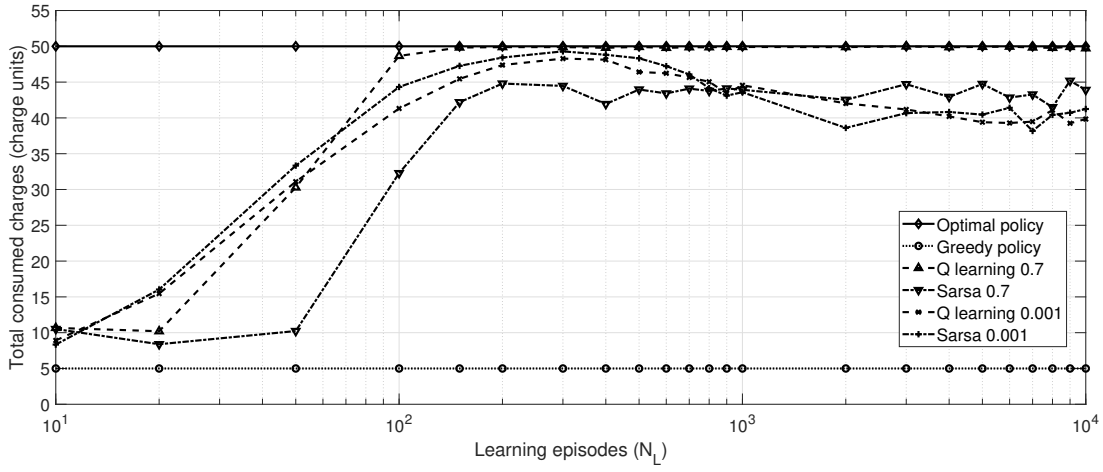

Figure 5.5: Total consumed charges vs learning episodes with $B_{max} = 5$ and $T_B = 50$

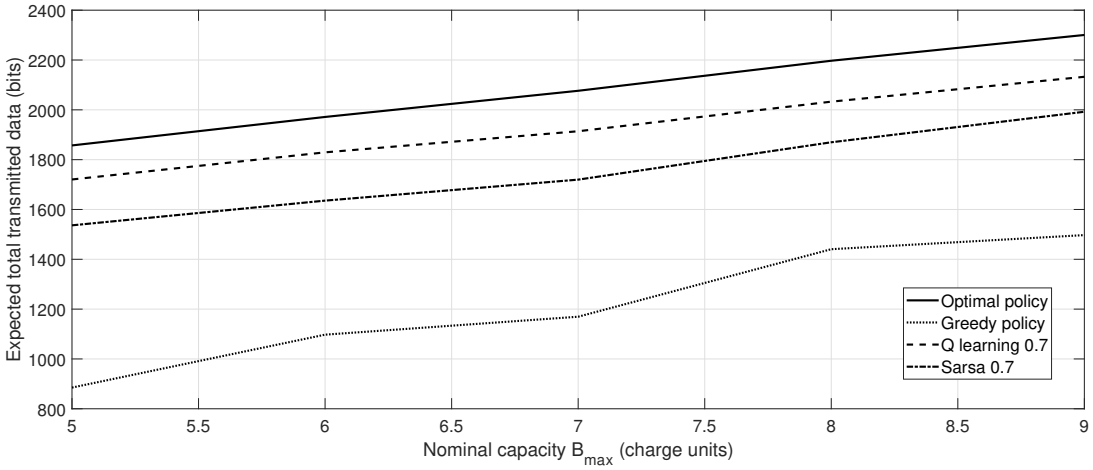

Figure 5.6: Expected total transmitted data vs B_{max} with $N_L = 1000$ and $T_B = 50$

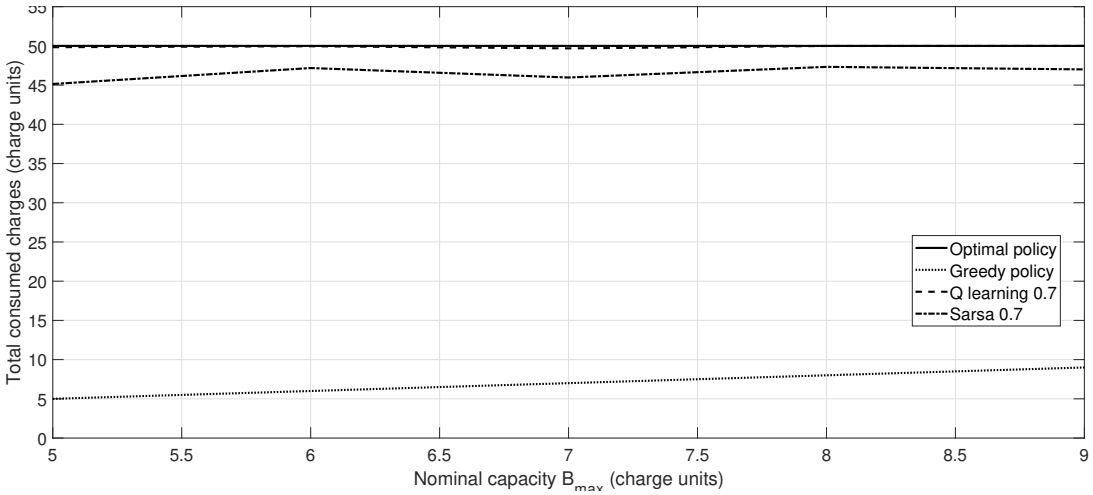

Figure 5.7: Total consumed charges vs B_{max} with $N_L = 1000$ and $T_B = 50$

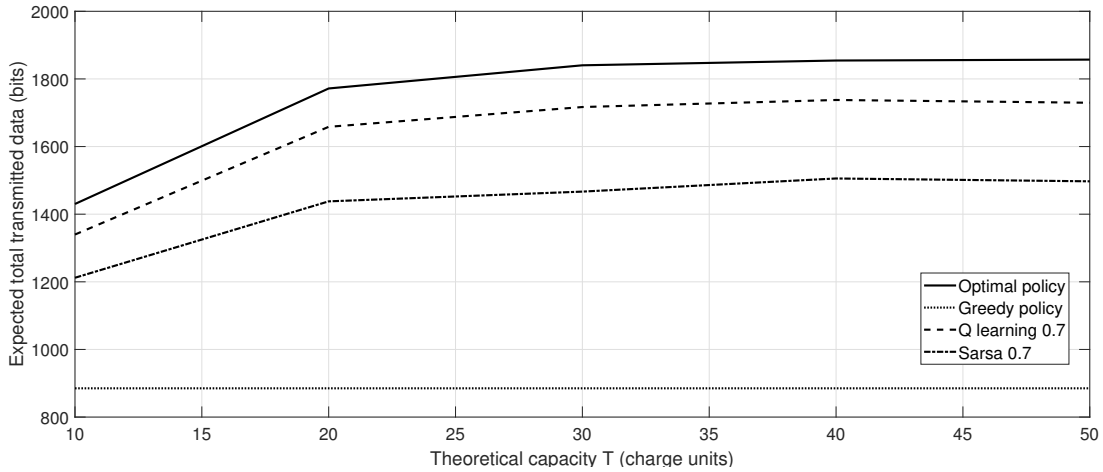

Figure 5.8: Expected total transmitted data vs T_B with $B_{max} = 5$ and $N_L = 1000$

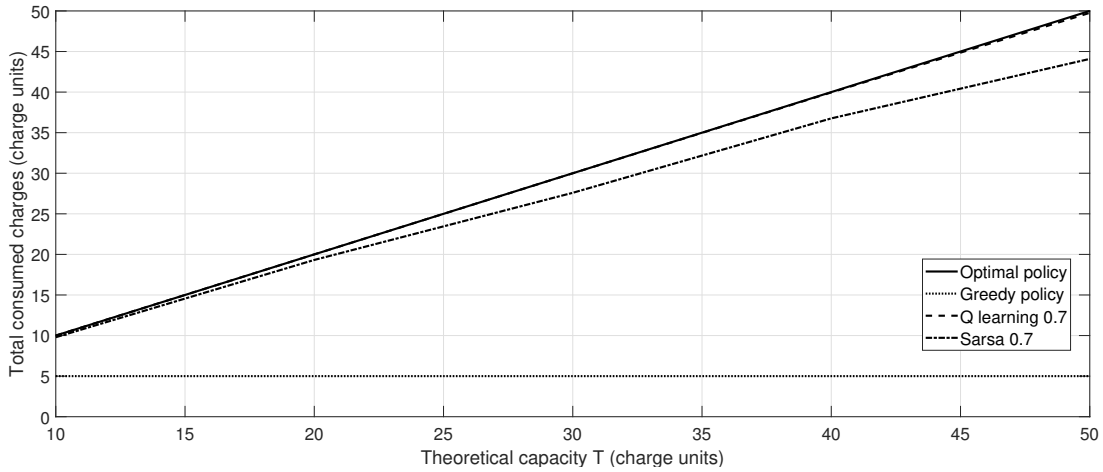

Figure 5.9: Total consumed charges vs T_B with $B_{max} = 5$ and $N_L = 1000$

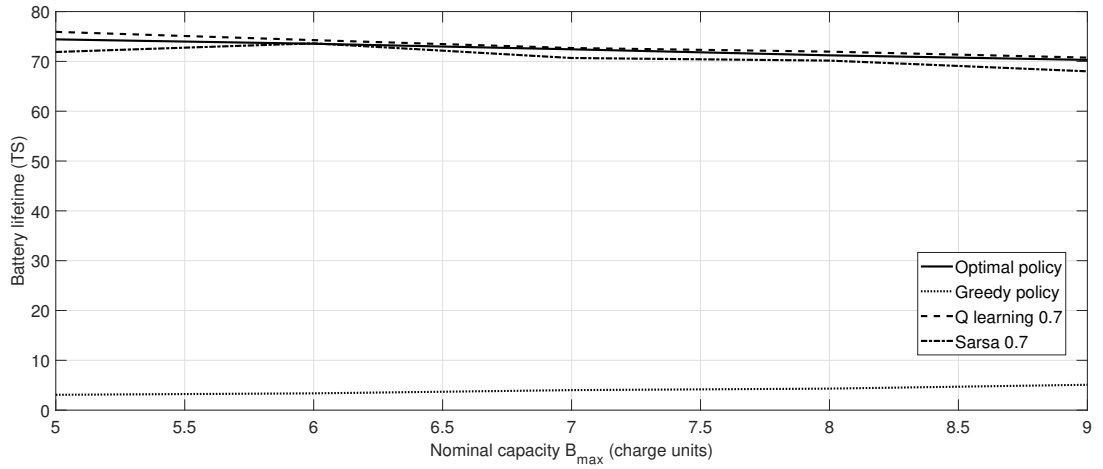

Figure 5.10: Battery lifetime vs B_{max} with $N_L = 1000$ and $T_B = 50$

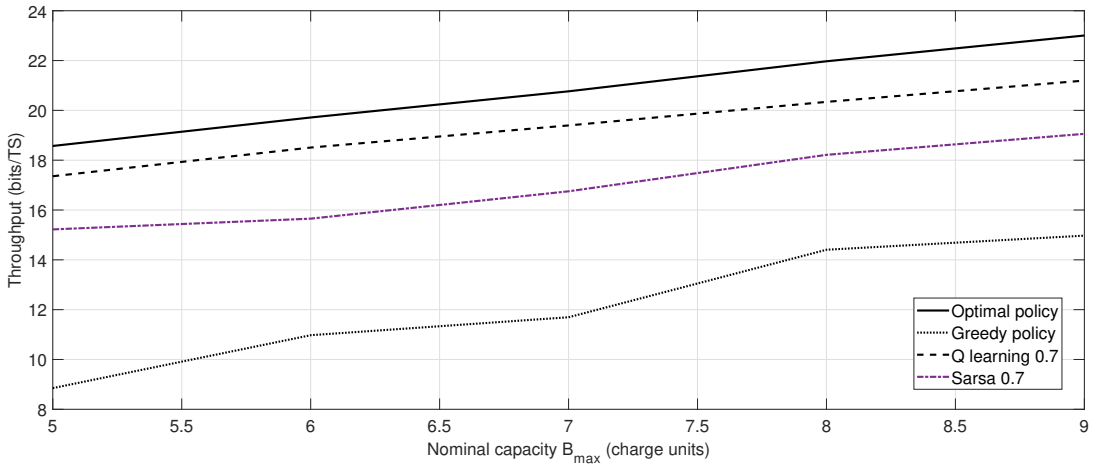

Figure 5.11: Throughput vs B_{max} with $N_L = 1000$ and $T_B = 50$

5.6 Partially known environment: a new Rapid Learning Approach model

On the one hand, conventional learning methods (e.g. Q-learning and Sarsa) operate by estimating the value of the action state pair during the learning phase. This is done considering an environment with dynamics completely unknown. Learning methods take a lot of time because they must perform many episodes to have sufficient knowledge of the behavior of the system [144, 145]. On the other hand, dynamic programming approaches (e.g. value iteration) require a complete knowledge of the behavior of system states, which is not at all verified in real cases. In several environments, some dynamics of the system may be partially known. In Table 5.1 we list different types of dynamics of the system that can be established in advance or are completely unknown.

Description	*Known*	*Unknown*
Deterministic	Battery charge state b_n	N/A
	Consumed theoretical charge capacity thc_n	N/A
Completely unknown	N/A	Data packet size d_n
	N/A	Channel state h_n

Table 5.1: Classification of the dynamics of the system states

Actually, as soon as the decision to transmit a data packet is made, the power control unit set the number of units to be consumed and once the transmission performed, the next battery state and the consumed theoretical capacity can then easily be deduced.
We define an intermediate state $\tilde{S}$ to describe the state of the system after the so-called optimal action has been taken and well before the unknown dynamics can take place. This mechanism has already been adopted in previous publications, see [146, 147]. Based on optimization subareas of system states, the relationship established at time n between the intermediate state $\tilde{S}_n$, the system state $S_n = (b_n, thc_n, d_n, h_n)$ the action a_n ($\in \{0, 1\}$) and the next system state S_{n+1} is given as follows:

- The intermediate system state is $\tilde{S}_n = (\tilde{b}_n, t\tilde{h}c_n, d_n, h_n) = (b_n - c_n, thc_n - c_n, d_n, h_n)$,
- The next system state $S_{n+1} = (b_{n+1}, thc_{n+1}, d_{n+1}, h_{n+1}) = (\tilde{b}_n, t\tilde{h}c_n, d_{n+1}, h_{n+1})$.

Where c_n is the number of charge units extracted from the battery and used for transmission purposes in case of wireless card turned on ($a_n = 1$). Otherwise, $c_n = 0$.
The intermediate state is a fictitious one for which we assume that the values for packet sizes and channel states, after performing an action, remain unchanged. It takes into account only the known parts of the studied system when transition from system state S_n to the state S_{n+1} occurs after making the decision a_n. Depending on the consumption of charge units for the transmission requirements, the updating of the battery levels and the consumed theoretical capacity is first operated at this stage. By introducing the notion

of intermediate state, we can split the transition probability function into two separated parts a known and unknown ones. The known part informs about the transition from the current state to the intermediate state, i.e. $S_n \rightarrow \tilde{S}_n$, while the unknown part governs the transition from the intermediate state to the next state $\tilde{S}_n \rightarrow S_{n+1}$ as established in Equation 5.17.

$$p(S_n, a_n, S_{n+1}) = \sum_{\tilde{S}} p_u(\tilde{S}, a_n, S_{n+1}) p_k(S_n, a_n, \tilde{S}) \tag{5.17}$$

Where the index k and u respectively denote the the known and unknown parts of the transition probability function. In this case, the reward is generated during the transition from the current state to the intermediate state according to the value of the action taken from the policy followed by the wireless sensor. The phase corresponding to the transition from intermediate to the next state can also generate an additional reward as established in Equation 5.18.

$$R(S_n, a_n, S_{n+1}) = R_u(\tilde{S}, a_n, S_{n+1}) + R_k(S_n, a_n, \tilde{S}) \tag{5.18}$$

In our case, we notice that the unknown parts of the next system state do not depends on the action having been taken and thus do not contribute to the production of the reward as given in Equations 5.19 and 5.20.

$$p_u(\tilde{S}, a_n, S_{n+1}) = p_u(\tilde{S}, S_{n+1}) \tag{5.19}$$

$$R_u(\tilde{S}, a_n, S_{n+1}) = R_u(\tilde{S}, S_{n+1}) = 0 \tag{5.20}$$

However, this approach can easily be extended to cases where the unknown part depends on both the intermediate state and the action undertaken in the first phase. In this case an exploration of all possible actions will be necessary to determine the optimal policy. The known and unknown transition probability functions can therefore be expressed as in Equations 5.21 and 5.22.

$$\begin{aligned} p_k(S_n, a_n, \tilde{S}_n) =& p_b(b_n, a_n, \tilde{b}_n) p_t(thc_n, a_n, \tilde{thc}_n) \\ & I(\tilde{d}_n = d_n) I(\tilde{h}_n = h_n) \end{aligned} \tag{5.21}$$

$$\begin{aligned} p_u(\tilde{S}_n, S_{n+1}) =& p_d(\tilde{d}_n, d_{n+1}) p_h(\tilde{h}_n, h_{n+1}) \\ & I(b_{n+1} = \tilde{b}_n) I(thc_{n+1} = t\tilde{h}c_n) \end{aligned} \tag{5.22}$$

Where the operator I(.) is the indicator function that takes the value 1 if the argument in parentheses is true and the value 0 otherwise. The reward generated by the known part at each iteration is given in Equation 5.23.

$$R_k(S_n, a_n) = a_n \times d_n \tag{5.23}$$

5.6.1 Dynamic programming based concept for Rapid Learning Algorithm

Before proceeding with the description of the learning algorithm, we first define the state value function $V(S_n)$. This function will play the similar role as the action-state value function for the Q-learning algorithm. The optimal state value function, denoted $v^*_{S_r}$, is computed for each fixed pair (d_n, h_n) as a reduced set of system states S_r. We have to go through all the possible combinations for these two elements in order to finally find v^* for all possible system states. $v^*_{S_r}$ is computed according to Equation 5.24.

$$\begin{aligned} v^*_{S_r}(S_n) =& \max_{a_i} \\ & \left\{ R_k(S_n, a_i) + \gamma \sum_{S_{red} \in S_r} p_k(S_n, a_i, S_r) V_{S_r}(S_{red}) \right\} \end{aligned} \tag{5.24}$$

Where S_{red} is the possible future state in the reduced set of system states S_r. S_r is based on the pair (packet size, channel state) as given in the current system state S_n. Having the optimal value $v^*_{S_r}$, the optimal policy can be directly obtained by Equation 5.25.

$$\begin{aligned} \pi^*_{S_r}(S_n) =& \operatorname*{argmax}_{a_i} \\ & \left\{ R_k(S_n, a_i) + \gamma \sum_{S_{red} \in S_r} p_k(S_n, a_i, S_{red}) V(S_{red}) \right\} \end{aligned} \tag{5.25}$$

The following proposition proves that $\pi^*_{S_r}(S_n)$, as defined in 5.25, and $\pi^*(S_n)$, as defined in 5.12 are equivalent.

Proposition 1: $\pi^*_{S_r}(S_n)$ and $\pi^*(S_n)$ are equivalent.

Proof: To show that $\pi^*_{S_r}(S_n)$ and $\pi^*(S_n)$ are equivalent, we split the total set of system states S on subsets S_{r_i} with $i \in \{1, 2, ..., N_D \times N_H\}$. In our case, $N_D = 2$ and $N_H = 2$, therefore $i \in \{1, 2, 3, 4\}$ as given in Equation 5.26.

$$
\begin{aligned}
\pi^*_{S_{r_1}}(S_n) =& \underset{a_i}{\text{argmax}} \\
& \left\{ R_k(S_n, a_i) + \gamma \sum_{S_{red} \in S_{r_1}} p_k(S_n, a_i, S_{red}) V_{S_{r_1}}(S_{red}) \right\} \\
& = \underset{a_i}{\text{argmax}} \\
& \left\{ R_k(S_n, a_i) + \gamma \left(\sum_{S_{red} \in S_{r_1}} p_k(S_n, a_i, S_{red}) V_{S_{r_1}}(S_{red}) \right.\right. \\
& \sum_{S_{red} \in S_{r_2}} p_k(S_n, a_i, S_{red}) V_{S_{r_2}}(S_{red}) + \\
& \sum_{S_{red} \in S_{r_3}} p_k(S_n, a_i, S_{red}) V_{S_{r_3}}(S_{red}) + \\
& \left.\left. \sum_{S_{red} \in S_{r_4}} p_k(S_n, a_i, S_{red}) V_{S_{r_4}}(S_{red}) \right) \right\} \\
& = \underset{a_i}{\text{argmax}} \\
& \left\{ R_k(S_n, a_i) + \gamma \sum_{S' \in S} p_k(S_n, a_i, S') V(S') \right\} \\
& = \pi^*(S_n)
\end{aligned}
\tag{5.26}
$$

Similarly for the cases of $\pi^*_{S_{r_2}}(S_n)$, $\pi^*_{S_{r_3}}(S_n)$ and $\pi^*_{S_{r_4}}(S_n)$ which can all be proved equivalent to $\pi^*(S_n)$. In other words, $\pi^*(S_n)$ is the concatenation of the set of $\pi^*_{S_{r_i}}(S_n)$ since each covers a reduced set of S.

Proposition 1 is very important because it allows us to use the $\pi^*_{S_{r_i}}(S_n)$ to learn the optimal policy $\pi^*(S_n)$.

5.6.2 The proposed Rapid Learning Algorithm

While Q-learning algorithm learns the action-state-value function to approach its optimal value $q^*(S_n)$, the proposed method focus on state-value function and directly calculates its optimal value $v^*(S_n)$. However, even if the elements of the optimal vector v^* are computed step by step while fixing the known parts of the system, the whole process converges quickly to a robust estimate of the global optimal policy π^*. The learning process includes exploring unknown parts of the system. Actually, for each combination of data packet size and channel status identified, a dynamic optimization is conducted to update the system state value vector $v(S_n)$. Schematically, the proposed method explores the episodes of learning phases as in Figure 5.12.

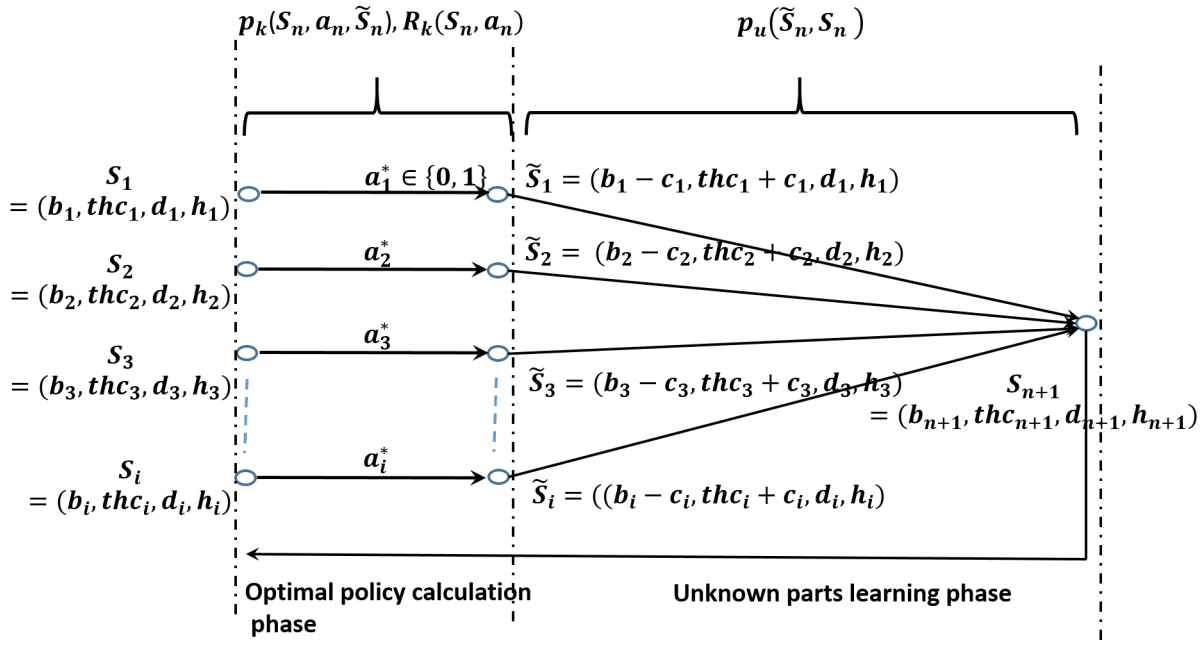

Figure 5.12: The proposed rapid learning process description

The Rapid Learning Algorithm is given as in Algorithm 13. In the Value iteration case, the calculation complexity based on the entire system state S were bounded by $O(\frac{2^{N_S}}{N_S})$. However, in the RLA case, the system model becomes S_r and the complexity is therefore reduced to be bounded by $O(\frac{2^{N_{S_r}}}{N_{S_r}})$. As the ratio of the two system dimensions is of $\frac{N_S}{N_{S_r}}$, the complexity is then reduced by a factor of $\frac{2^{N_D N_H}}{N_D N_H}$.

Algorithm 13 Proposed Rapid-learning algorithm

1. ***Initialization step:***

$n = 0$, read the initial system state $S_n = (b_0, thc_0, h_0, d_0)$

Initialize $v = zeros(N_S, 1)$ and $\pi = zeros(N_S, 1)$ set $i = 0$ and set $W = \Phi$ and $c = (h_n, d_n)$

2. ***Learning step:***

if $c \notin W$ **then**

Set the sub system states $S_r = \{B \times T, c\}$

$\epsilon \leftarrow 0.001$

while $\Delta > \epsilon$ **do**

$\Delta \leftarrow 0$

$temp \leftarrow V$

for each state $S_m \in S_r$ **do**

$$v(S_m) \leftarrow \max_{a \in A} \sum_{\forall S_j \in S_r} p(S_m, a, S_j)[R_k(S_m, a, S_j) + \gamma v(S_j)]$$

$$\pi(S_m) \leftarrow \underset{a \in A}{\operatorname{argmax}} \sum_{\forall S_k \in S_r} p(S_m, a, S_j)[R_k(S_m, a, S_j) + \gamma V(S_j)]$$

end for

$\Delta \leftarrow max(\Delta, \| v(S_m) - temp \|_\infty)$

end while

end if

Apply action $a_n = \pi(S_n)$

Observe the system states experience results $S_n \rightarrow a_n \rightarrow R_k \rightarrow b_{n+1}, thc_{n+1}$

$c \rightarrow W$

read the next data packet size d_{n+1} and channel state h_{n+1}

set $n \leftarrow n + 1$,

set $c = (h_n, d_n)$ go to 2

End of the episode

5.7 Numerical Results: 4th contribution

In this section the results of the publication titled "Rapid learning optimization approach for battery recovery-aware embedded system communications" [148] are presented and discussed.

We define all the features of the considered embedded system that supports the battery recovery case study. For comparison purposes by simulation we consider a greedy algorithm where the wireless card is always switched on where and the transmitter sends continually the data packets it receives, which means that the decision being permanently made in that case is $(a_n = 1, n = 1 : \infty)$ all the time. The performances obtained from the Value iteration, Rapid Learning approach, Q-learning and Greedy algorithms are compared both in terms of the amount of data transmitted and the consumed theoretical capacity of the battery.

We introduce an updating of the system simulation platform having been used in Section 2, with an additional constraint for the nominal battery capacity. Now we consider also the limitation of the nominal capacity as the charge units are consumed. Approaching the theoretical capacity, the updating rules is then bounded by the quantity of charges still available in the electrolyte $(T_B - thc_n)$, where thc_n is the cumulative quantity of the consumed charge units of the theoretical capacity. Battery charge units quantities b_n are now updated according to Equation 5.27 at the end of each iteration (which replace the Equation 5.4 used in Section 6).

$$B_{n+1} = \begin{cases} B_n - E_n^{T_S} & \text{if } a_n = 1 \\ min\left(B_n + e, B_{max}, T_B - thc_n\right) & \text{if } a_n = 0 \end{cases} \tag{5.27}$$

We consider episodes with duration's length of $100 \times T_S$, given that a discounted factor γ of 0.9 concentrates more than 99% of the whole significant amount of transmitted data in the interval 1 to 100. 2000 episodes are generated to determine an average of the results. These episodes focus primarily on varying the size of data packets and channel states that depend on their proper transition probabilities.

In Fig. 5.13 and Fig. 5.14, for a theoretical capacity value of $T_B = 20$ charge units

and nominal capacity $B_{max} = 5$, Rapid Learning Algorithm reaches 99.8% of the optimal expected total transmitted data and consumes up to 94.5% of the available theoretical capacity which is equal to 96.16% of the optimal consumed capacity by the $37th$ iteration. Q-learning needs to run over than 200000 iterations (Q-learning 2×10^5) to reach 87.7% of the optimal expected total transmitted data and to consume only 60.5% of the available theoretical capacity which is equal to 61.6% of the optimal consumed capacity.

In Fig. 5.15, we notice that the optimal policy generated by the Value iteration Algorithm sends an average amount of data of up to 1297.9 bits per episode of 100 iterations. The Rapid Learning Algorithm reaches an average amount of transmitted data of 1295.5 bits, while Q-learning achieves a transmission performance of 1138.9 transmitted bits. The Greedy algorithm arrives at the bottom position with the lowest performance by sending an average amount of data of only 535.6 transmitted bits.

In Fig. 5.16, the optimal Algorithm achieves an average consumption of 19.6 charge units from the 20 available theoretical capacity T_B. The Rapid Learning Algorithm reaches an average amount of consumed charge units of 18.9, while Q-learning achieves a performance of 12.1 consumed charge units. The greedy algorithm arrives at the last position, achieving the worst performance when it consumes a quantity of units of charge of only 5 units leaving 15 units as unused capacity.

In Fig. 5.17, optimal algorithm manages to avoid the battery trapping state as long as charge units are still available in the battery's electrolyte. The Rapid learning algorithm takes the same challenge and tries to perform an optimal use of the capacity of the battery as well as the Q-learning algorithm. For those three algorithms the nominal capacity of the battery is kept greater than 0 for all the episode duration leaving an average quantity of 0.332 , 0.0275 and 0.059 charge units respectively for the optimal, rapid and Q-learning algorithms by the end of the episode. For the greedy algorithm, the trapping state is quickly reached after an average number of 3 transitions only, making the battery unusable while many units of charge are still unused.

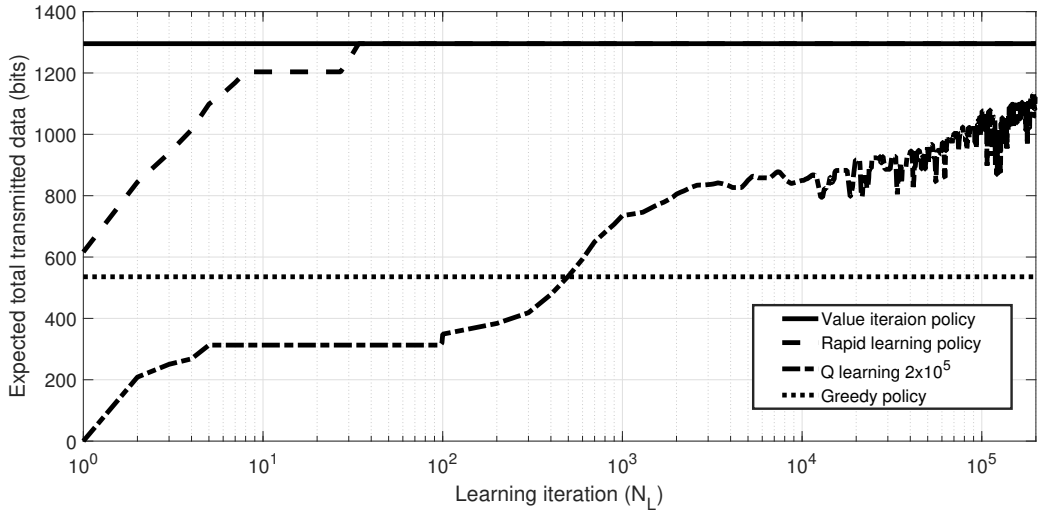

Figure 5.13: Expected total transmitted data vs N_L with $T_B = 20$ and $B_{max} = 5$

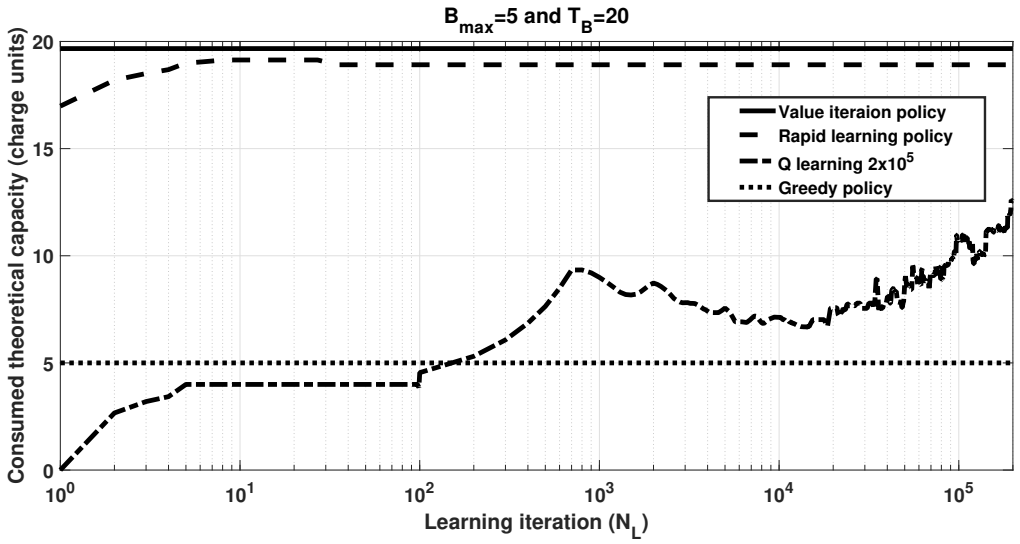

Figure 5.14: Consumed theoretical capacity vs N_L with $T_B = 20$ and $B_{max} = 5$

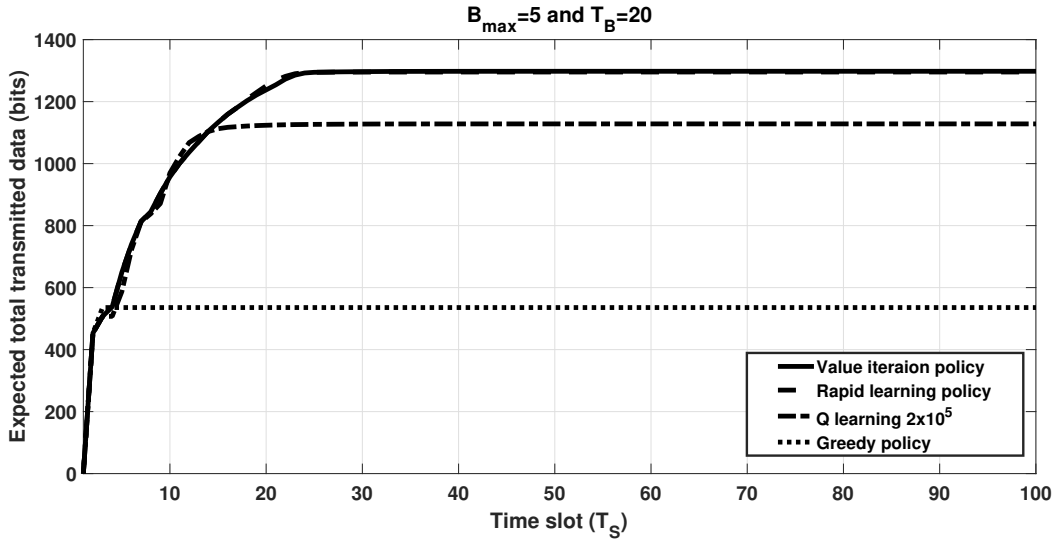

Figure 5.15: RLA comparison results for Expected total transmitted data vs time slots given $B_{max} = 5$ and $T_B = 20$

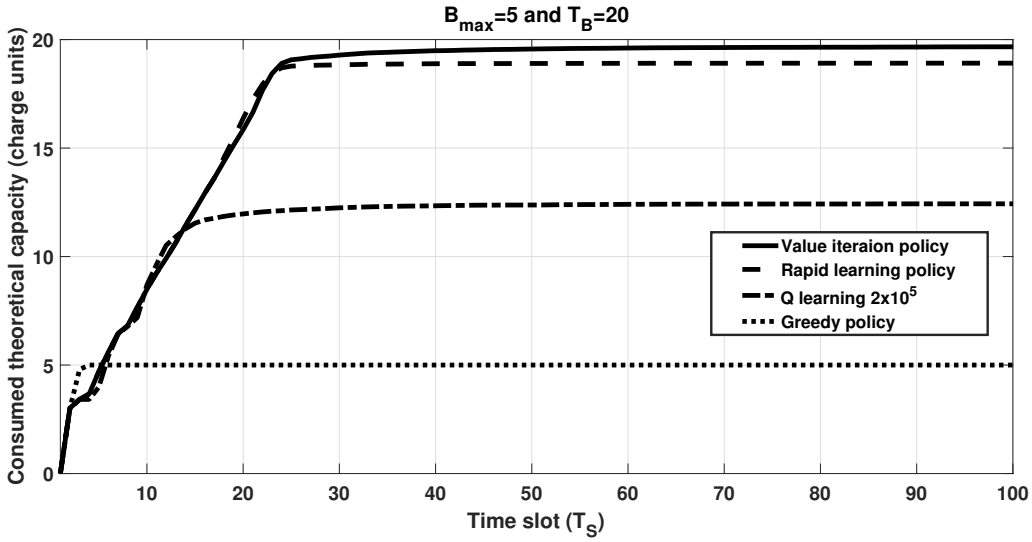

Figure 5.16: RLA comparison results for Consumed theoretical capacity vs time slots given $B_{max} = 5$ and $T_B = 20$

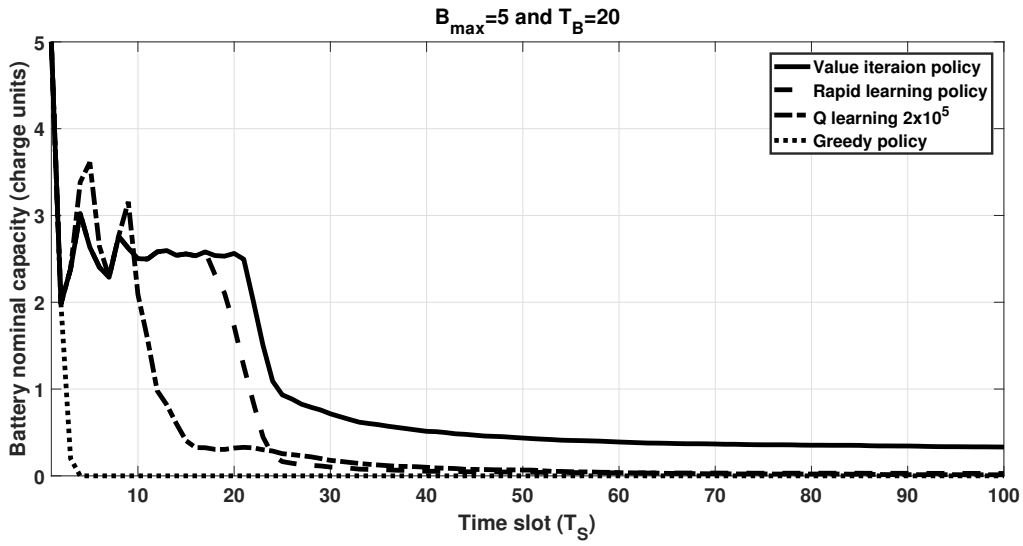

Figure 5.17: RLA comparison results for Nominal capacity vs time slots given $B_{max} = 5$ and $T_B = 20$

5.8 Conclusion

In this chapter, we have addressed the optimization problem of the battery recovery-aware embedded system communications. First we consider that prior stochastic information is provided for the system and we apply Value Iteration approach to generate an optimal policy. Then we introduce the algorithms from RL domain that were Q-learning and Sarsa to deal with an absence of such information as in realistic situations. We proved that when the transmission policy is learned by the system, the obtained cumulative rewards are closer to those of the Value Iteration for high and low ϵ-rate values. We also deduce that the Q-leaning algorithm is more suitable with high exploration appetite $\epsilon = 0.7$ since it offers high expected cumulative rewards, long battery lifetime an high throughput performances. In addition, we have proved that according to the battery choice, high theoretical battery capacities is not optimal as a saturation phenomenon can occur. However, high nominal capacities enhance the performance of the system in terms of cumulative rewards.

We have also proposed a new rapid learning algorithm, compatible with the optimization of a communication taking into account the recovery of the battery in the case of embedded systems. Actually, we have noticed that, among the information available on the studied

system, the Q-learning approach did not take into account the battery transitions and consumed charges levels that we can predict after each data packet transmission. This observation led us to propose the third type of scenario where we have a partial knowledge of the studied system. We then introduce the rapid learning algorithm that has worked much faster than Q-learning, as shown by the results of the simulation taking into account the partially known information.

Thanks to these algorithms, the investment costs of embedded systems can also be considerably reduced, e.g. the investment in batteries for the greedy algorithm costs 4 times more than in the case of rapid learning. In addition, less battery waste is produced, which better protects the environment and mitigates the impact of this type of waste on the planet.

Chapter

6 Conclusion and future work

This chapter summarizes the main contributions of this thesis and concludes by presenting the future directions of the research.

Summary of Research

This thesis work focused on the optimization of energy in embedded systems in order to make good use of batteries before their exhaustion. The performance of the system evaluated is both the amount of data processed and transmitted and the longevity of the whole embedded system in relation to the useful duration of operation. We have considered peer-to-peer communication between intelligent sensors or between intelligent sensors and base stations (sinks) to which the data packets are destined. Embedded systems were supposed to be powered by primary or secondary batteries. In the case of primary cells, we focused on the phenomenon of energy recovery in the electrolyte to draw new charging units and thus renew the nominal capacity of the battery. This approach has greatly contributed to increasing the amount of data successfully transmitted to the destination while ensuring optimal system lifetime. In the case of the use of secondary batteries, recharging based on the energy harvesting from the surrounding environment has been considered in order to maximize the performance of the embedded system communications.

Two scenarios for the case of the secondary battery and three scenarios for the case of the primary battery were dealt with in this thesis regarding prior availability of information on the data transmission environment. In the first scenario, the stochastic model of the embedded system environment was assumed to be completely known and the optimization of the sensor transmission policy was computed using algorithms from the field of dynamic programming such as policy iteration and value iteration. In the second scenario, it was assumed that no information was known in advance, so it is up to the system itself to conduct a learning phase before deriving the optimal control policy by using algorithms from the reinforcement learning domain such as Q-learning, Sarsa and Q-Sarsa.
A third scenario was considered, which was applied for the first time in the case of a primary battery with recovery effect, where it was assumed that the information on the simulation environment could be partially known. What we were able to verify in the case of the embedded system model considered for the studies conducted. We have therefore proposed a new algorithm that exploits the known parts of the simulation environment and performs a short learning phase to fill the information gap left by the unknown parts. With the rapid learning algorithm, interesting results have been obtained, associated with a significant reduction in the time needed to calculate the optimal policy.

Future work

The solutions proposed throughout this thesis can be improved in several ways. Some ideas that need to be investigated and developed further are given as follows:

- A time-varying recovery rate: The next problem complexity to deal with, in the case of recovery-aware battery powered system, is to consider the rate of recovery as timely varying. In fact, The rate of recovery is not constant during discharge, and in most systems the discharge current changes over time;

- A phase-dependent recovery rate: A phase-dependence behavior of the recovery rate may also be considered to enhance the model being used in order to make more realistic. A phase number can be considered equivalent to the number of charge units that has been consumed. In fact, when more charge units have been

consumed, the phase number increases and this causes the probability of recovery to decrease.

These behavioral characteristics relative to the recovery rate lead us to search for new battery simulation models to take into account the different dependencies mentioned above. As a first solution, the generalized pulsed discharge model proposed by Chiasserini and Rao could partly respond to these concerns at the cost of an increasing complexity of the calculations to be carried out.

Other learning algorithms can also be proposed and tested for their relevance compared to the results obtained in this thesis such as Deep learning, Deep reinforcement learning, Deep Q-learning, etc.

References

[1] K. Grover, D. Kahali, S. Verma ans B. Subramanian, "WSN-based system for forest fire detection and mitigation", In Emerging Technologies for Agriculture and Environment, pp. 249-260, Springer, Singapore, 2020.

[2] A. Aksamovic, M. Hebibovic and D. Boskovic, "Forest fire early detection system design utilising the WSN simulator", In 2017 XXVI International Conference on Information, Communication and Automation Technologies (ICAT), pp. 1-5, IEEE, 2017.

[3] J. Lee, J. Kim, D. Kim, P. K. Chong, J. Kim, and P. Jang, "Rfms: Realtime flood monitoring system with wireless sensor networks", In Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on, pages 527-528. IEEE, 2008.

[4] L. Huang and S. Cheng, "Unmanned monitoring system of rivers and lakes based on wsn", In Systems and Informatics (ICSAI), 2012 International Conference on, pages 495-498. IEEE, 2012.

[5] T. Nagayama, M. Ushita, and Y. Fujino, "Suspension bridge vibration measurement using multihop wireless sensor networks", Procedia Engineering, 14, 761-768, 2011.

[6] K. Mekki, A. Zouinkhi, W. Derigent, E. Rondeau, A. Thomas, and M. N. Abdelkrim. Usee: A uniform data dissemination and energy efficient protocol for communicating materials. Future Generation Computer Systems, 56:651-663, 2016.

[7] W.A. Geoffrey, J. Jeff, R. Mario, L. Jonathan, and W. Matt, "Monitoring volcanic eruptions with a wireless sensor network", In Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on, pages 108-120. IEEE, 2005.

[8] M. V. Ramesh, "Wireless sensor network for disaster monitoring", Wireless Sensor Networks: Application-Centric Design, 2010.

[9] I. Bennis, O. Zytoune, D. Aboutajdine and H. Fouchal, "Low energy geographical routing protocol for wireless multimedia sensor networks". In Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International (pp. 585-589), IEEE, (July, 2013).

[10] I. Dbibih, O. Zytoune and D. Aboutajdine, "On/off markov model based energy-delay aware mac protocol for wireless sensor network". Wireless personal communications, 78(2), 1143-1155, (2014).

[11] O. Zytoune and D. Aboutajdine, "Energy usage analysis of digital modulations in wireless sensor networks with realistic battery model". Wireless Networks, 22(8), 2713-2725, (2016).

[12] M. K. Sharma and C. R. Murthy, "On the Design of Dual Energy Harvesting Communication Links With Retransmission". IEEE Transactions on Wireless Communications, v. 16 (6), June 2017.

[13] O. Kanoun (Ed.), "Energy Harvesting for Wireless Sensor Networks: Technology, Components and System Design", Walter de Gruyter GmbH and Co KG, 2018.

[14] K. Ashton, "That internet of things thing". In the real world, things matter more than ideas", Expert view. RFID J. (2009). http://www.rfidjournal.com/articles/view?4986.

[15] S. Kuyoro, F. Osisanwo, and O. Akinsowon, "Internet of Things (IoT): An Overview", In 3rd International Conference on Advances in Engineering Sciences and Applied Mathematics (pp. 53-58), (2015, March).

[16] R. Want, B.N. Schilit, S. Jenson, "Enabling the Internet of Things", Computer 48 (2015) 28-35.

[17] National Intelligence Council, Disruptive technologies global trends 2025. Six technologies with potential impacts on US interests out to 2025 (2008).

[18] D. Evans, "The internet of things: How the next evolution of the internet is changing everything", CISCO white paper, 1-11, (2011).

[19] CASAGRAS, CASAGRAS EU project final report, https://docbox.etsi.org/zArchive/TISPAN/Open/IoT/low%20resolution/www .rfidglobal.eu%20CASAGRAS%20IoT%20Final%20Report%20low%20resolution.pdf.

[20] M.T. Lazarescu, "Design of a WSN platform for long-term environmental monitoring for IoT applications", IEEE J. Emerging Sel. Top. Circuits Syst. 3 (2013) 45-54.

[21] P. Eugster, V. Sundaram, X. Zhang, "Debugging the Internet of Things: the case of wireless sensor networks", IEEE Software 32, (2015), 38-49.

[22] L. Mainetti, L. Patrono, A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: a survey", in: Software, Telecommunications and Computer Networks (SoftCOM), 2011, 19th International Conference on, 2011, 2011, pp. 1-6.

[23] S. Hong, D. Kim, M. Ha, S. Bae, S.J. Park, W. Jung, et al., "SNAIL: an IP-based wireless sensor network approach to the internet of things", IEEE Wireless Commun., 17, (2010), 34-42.

[24] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a service model for smart cities supported by Internet of Things", Trans. Emerg. Telecommun. Technol., vol. 25, no. 1, pp. 81-93, 2014.

[25] S. Deering and R. Hinden, "Internet protocol, version 6 (ipv6) specification",The Internet Engineering Task Force (IETF), Tech. Rep., 1998.

[26] J. A. Manrique, J. S. Rueda-Rueda, and J. M. Portocarrero, "Contrasting internet of things and wireless sensor network from a conceptual overview", In 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) (pp. 252-257). IEEE, (2016, December).

[27] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized protocol stack for the internet of (important) things", IEEE communications surveys and tutorials, 15(3), 1389-1406, 2012.

[28] C. Bormann, A. P. Castellani and Z. Shelby, "CoAP : An Application Protocol for Billions of Tiny Internet Nodes", IEEE Internet Computing, vol. 16, no 2, 2012, p. 62-67.

[29] Z. Shelby, K. Hartke, C. Bormann, and B. Frank, "Constrained Application Protocol (CoAP)", IETF CoRE Working Group, Feb. 2011.

[30] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview,Assumptions, Problem Statement, and Goals", IETF, RFC 4919, Aug. 2007.

[31] Z. Shelby and C. Bormann, "6LoWPAN: The Wireless Embedded Internet", Wiley Series on Communications Networking and Distributed Systems. John Wiley and Sons, 2010.

[32] T. Winter, P. Thubert, A. Brandt, T. Clausen, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, and JP. Vasseur, "RPL: IPv6 Routing Protocol for Low power and Lossy Networks", IETF ROLL working group, Mar. 2011.

[33] IEEE std. 802.15.4. Part. 15.4: Wireless Medium Access Control (MAC) and PhysicalLayer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Standard for Information Technology, Sep. 2006.

[34] D. Evans, "The internet of things how the next evolution of the internet is changing everything", in CISCO White Paper (2011).

[35] J. McCarthy, M. Minsky, N. Rochester, C. Shannon, "A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence". Archived from the original on 26 August 2007. Retrieved 30 August 2007.

[36] J. McCarthy, "What is Artificial Intelligence?," 2001. [Online]. Available: http://lidecc.cs.uns.edu.ar/ grs/InteligenciaArtificial/whatisai.pdf. [Accessed: 26-Apr-2016].

[37] V. GarcÃa-DÃaz, J. Pascual-Espada, C. Pelayo G-Bustelo, and J. M. Cueva-Lovelle, "Towards a Standard-based Domain-specific Platform to Solve Machine Learning-based Problems," Int. J. Interact. Multimed. Artif. Intell., vol. 3, no. 5, p. 6, 2015.

[38] P. Simon, "Too Big to Ignore: The Business Case for Big Data", 1st ed. Wiley, 2013.

[39] https://www.gartner.com/smarterwithgartner/top-trends-on-the-gartner-hype-cycle-for-artificial-intelligence-2019/, visited on March, 18, 2020.

[40] M. Wernick, Y. Yang, J. Brankov, G. Yourganov, and S. Strother, "Machine Learning in Medical Imaging," IEEE Signal Process. Mag., vol. 27, no. 4, pp. 25-38, Jul. 2010.

[41] M. Mohri, A. Rostamizadeh, and A. Talwalkar, "Foundations of machine learning". MIT press, 2012.

[42] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning: Data Mining, Inference, and Prediction", 2nd ed., vol. 27, no. 2. New York, NY: Springer New York, 2009.

[43] N. Lange, C. M. Bishop, and B. D. Ripley, "Neural Networks for Pattern Recognition," J. Am. Stat. Assoc., vol. 92, no. 440, p. 1642, Dec. 1997.

[44] X. Zhu, "Semi-Supervised Learning Literature Survey," Sci. York, pp.1-59, 2007.

[45] J. A. Hertz, A. S. Krogh, R. G. Palmer, and A. S. Weigend, "Introduction to the Theory of Neural Computation", vol. I, no. June. Basic Books, 1991.

[46] D. Hughes, J. Ueyama, E. Mendiondo, N. Matthys, W. Horré, S. Michiels, C. Huygens, W. Joosen, K. L. Man, and S. Guan, "A middleware platform to support river monitoring using wireless sensor networks", Journal of the Brazilian Computer Society, 17(2), 85-102, 2011.

[47] W-A. Geoffrey, J. Jeff, R. Mario, L. Jonathan, and W. Matt, "Monitoring volcanic eruptions with a wireless sensor network", In Wireless Sensor Networks, 2005. Proceeedings of the Second European Workshop on, pages 108-120. IEEE, 2005.

[48] M. V. Ramesh, "Wireless sensor network for disaster monitoring", Wireless Sensor Networks: Application-Centric Design, 2010.

[49] V. Parashar, B. Mishra, and G. S. Tomar, "Energy Aware Communication in Wireless Sensor Network: A Survey", Materials Today: Proceedings, 29, 512-523, 2020.

[50] V. Malhotra A. Stritter C. Talarico, J.W. Rozenblit, "A new framework for power estimation of embedded systems", Computer, 38(2):71-78, 2005.

[51] S. Nikolaidis and T. Laopoulos, "Instruction-level power consumption estimation of embedded processors for low-power applications", Journal on Computer Standards and Interfaces, 24(2):133-137, 2002.

[52] M. Kocakulak, and I. Butun, "An overview of Wireless Sensor Networks towards internet of things", In 2017 IEEE 7th annual computing and communication workshop and conference (CCWC), pp. 1-6, IEEE, 2017.

[53] W. Kaiser G. Pottie, "Wireless integrated network sensors", Communication of ACM, 43(5):51-58, May 2000.

[54] M. B. Srivastava A. Kansal, "An environmental energy harvesting framework for sensor networks", In Proceedings of the 2003 International Symposium on Low Power Electronics and Design, ISLPED '03, pages 481-486, 2003.

[55] A. Montoya, D. C. Restrepo, , D. A. Ovalle, "Artificial intelligence for wireless sensor networks enhancement", Smart Wireless Sensor Networks, 73-81,2010.

[56] D. CRULLER, D. Estrin and M. Srivastava, "Overview of sensor networks", Computer, 37(8): 41-49, 2004.

[57] E. V. Denardo,"Contraction mappings in the theory underlying dynamic programming", SIAM Review, 9:165-177, 1967.

[58] T. Jaakkola, M. I. Jordan, and S. P. Singh. "On the convergence of stochastic iterative dynamic programming algorithms". Neural Computation, 6(6) :1185-1201, 1994.

[59] R. S. Sutton and A. G. Barto, ,"Reinforcement learning: an introduction". MIT Press, 2018.

[60] R. Sutton, "Learning to Predict by the Method of Temporal Differences", Machine Learning, vol. 3, Nbr 1, p. 9-44, 1988.

[61] C. Watkins, "Learning from Delayed Rewards", PhD thesis, Cambridge University, Cambridge, UK, 1989.

[62] D. Bertsekas,"Dynamic programming and optimal control", Athena Scientific, Belmont, MA, 1995.

[63] R.S. Sutton. "Temporal credit assignment in reinforcement learning". PhD Thesis, University of Massachusetts, Amherst, MA, 1984.

[64] R.S. Sutton. "Learning to predict by the methods of temporal difference". Machine Learning, 3, pp. 9-44, 1988.

[65] C. J. Watkins and P. Dayan. "Q-learning". Machine learning, 8(3-4), 279-292, 1992.

[66] D. Kamenetsky, "A comparison of neural network architectures in reinforcement learning in the game of othello". Technical report, School of Computing, University of Tasmania, 2005.

[67] S. Nissen, "Large scale reinforcement learning using q-sarsa(λ) and cascading neural networks", Unpublished masters thesis, Department of Computer Science, University of Copenhagen, KÃ¸benhavn, Denmark, 2007.

[68] R. McFarlane. A survey of exploration strategies in reinforcement learning. McGill University, http://www. cs. mcgill. ca/cs526/roger. pdf, accessed: April, 2018.

[69] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In Proceedings of the Seventh International Conference on Machine Learning, pp. 216-224. Morgan Kaufmann, San Mateo, CA, 1990.

[70] A. Pchelkin. Efficient exploration in reinforcement learning based on utile suffix memory. Informatica, 14(2), 237-250, 2003.

[71] M. A. Wiering. Explorations in efficient reinforcement learning (Doctoral dissertation, University of Amsterdam), 1999.

[72] A. C. Kolle. The nature of learning-a study of reinforcement learning methodology. Department of Computer Science, University of Copenhagen, 2003.

[73] J. Garche, C. K. Dyer, P. T. Moseley, Z. Ogumi, D. A. Rand and B. Scrosati, (Eds.), "Encyclopedia of electrochemical power sources" Newnes, 2013.

[74] Y. K. Tan and S. K. Panda, "Review of energy harvesting technologies for sustainable wireless sensor network", Sustainable wireless sensor networks, p. 15-43, 2010.

[75] T. Millward, "Power Supply Design. A Brief Tutorial," IEE Publisher, London 1998.

[76] G. Pistoia, "Batteries for portable devices", Elsevier, 2005.

[77] S. Roundy, D. Steingart, L. Frechette, P. Wright and J. Rabaey, "Power sources for wireless sensor networks", In European workshop on wireless sensor networks (pp. 1-17), Springer, Berlin, Heidelberg, January, 2004.

[78] A. R. Jha, "Next-generation batteries and fuel cells for commercial, military, and space applications", CRC Press, 2012.

[79] C. Knight, J. Davidson and S. Behrens, "Energy options for wireless sensor nodes", in Sensors, 8(12), 8037-8066, 2008.

[80] H.D. Linden, "Handbook of Batteries", 3rd ed., McGraw-Hill, New York 2002.

[81] C. Chiasserini and R. Rao, "Pulsed battery discharge in communication devices," in Proceedings of the 5th International Conference on Mobile Computing and Networking, pp. 88 - 95, 1999.

[82] M. Doyle, J. Newman, "Analysis of capacity-rate data for lithium batteries using simplified models of the discharge process," Jo. Applied Electt-ochem., vol. 27, no. 7, July 1997, pp. 846-856.

[83] B. Nelson, R. Rinehart, S. Varley, "Ultrafast pulse discharge and recharge capabilities of thin-metal film battery technology," 11th IEEE International Pulsed Power Conference, Baltimore, June 1997, pp. 636-641.

[84] Maxell, "Product technical data sheets," https://biz.maxell.com/en/primary-batteries/CR2032H-DataSheet-17e.pdf, visited on Februar 27, 2020.

[85] M. Doyle, T. F. Fuller, and J. Newman, "Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell," Journal of the Electrochemical Society, vol. 140, no. 6, pp. 1526 - 1533, 1993.

[86] T. F. Fuller, M. Doyle, and J. Newman, "Simulation and optimization of the dual lithium ion insertion cell," Journal of the Electrochemical Society, vol. 141, no. 1, pp. 1 - 10, 1994.

[87] T. F. Fuller, M. Doyle, and J. Newman, "Relaxation phenomena in lithium-ion-insertion cells," Journal of the Electrochemical Society, vol. 141, no. 4, pp. 982 - 990, 1994.

[88] T.S. DAO, C.P. VYASARAYANI and J. MCPHEE, "Simplification and order reduction of lithium-ion battery model based on porous-electrode theory", Journal of Power Sources, vol. 198, p. 329-337, 2012.

[89] M. Chen, G. A. Rincon-Mora, "Accurate Electrical Battery Model Capable of predicting runtime and I-V performance", IEEE transactions on Energy Conversion, Vol. 21, Issue 2, Pages 504 - 511, 2006.

[90] Y. Hu, S. Yurkovich, Y. Guezennec, "A technique for dynamic battery model identification in automotive applications using linear parameter varying structures", Control Engineering Practice, Vol. 7, Issue 10, Pages 1190-1201, 2009.

[91] M. Sarvi, M. Masoum, "A neural network model for Ni-Cd batteries", IEEE Universities Power Engineering Conference, Pages 1-5, 2008.

[92] A. Munkherjee, "Advances in battery management using neural networks and fuzzy logic", Cornell University, 2003.

[93] C. P. Zhang, J. Z. Liu and S. M. Sharkh, "Identification of Dynamic Model Parameters for Lithium ion batteries used in hybrid electric vehicles", International Symposium on Electric Vehicles (ISEV), 2009.

[94] M.R. Jongerden, B.R. Haverkort, "Which battery model to use?," in IET software, 2009, vol. 3, no 6, p. 445-457, 2009.

[95] S. C. Hageman, "Simple PSpice models let you simulate common battery types,"Electronic Design News, vol. 38, pp. 117 - 129, 1993.

[96] D. Rakhmatov and S. Vrudhula, "An analytical high-level battery model for use in energy management of portable electronic systems," in Proceedings of the International Conference on Computer Aided Design (ICCAD'01), 2001, pp. 488-493.

[97] J. Manwell and J. McGowan, "Lead acid battery storage model for hybrid energy systems," Solar Energy, vol. 50, pp. 399-405, 1993.

[98] J. Manwell and J. McGowan, "Extension of the kinetic battery model for wind/hybrid power systems," in Proceedings of the 5th European Wind Energy Association Conference (EWEC' 94), pp. 284-289, 1994.

[99] J. Manwell, J. McGowan, B.-G. E.I., S. W., and L. A., "Evaluation of battery models for wind/hybrid power system simulation," in Proceedings of the 5th European Wind Energy Association Conference (EWEC' 94), pp. 1182-1187, 1994.

[100] C. Chiasserini and R. Rao, "A model for battery pulsed discharge with recovery effect," in Wireless Communications and Networking Conference. IEEE Press, pp. 636 - 639, 1999.

[101] C. Chiasserini and R. Rao, "Improving battery performance by using traffic shaping techniques," IEEE Journal on Selected Areas in Communications, vol. 19, no. 7, pp. 1385 - 1394, 2001.

[102] C. Chiasserini and R. Rao, "Energy efficient battery management," IEEE Journal on Selected Areas in Communications, vol. 19, no. 7, pp. 1235 - 1245, 2001.

[103] J. Rabaey, J. Ammer, J. L. Da Silva and D. Patel. "PicoRadio: Ad-hoc wireless networking of ubiquitous low-energy sensor/monitor nodes". In Proceedings IEEE Computer Society Workshop on VLSI 2000. System Design for a System-on-Chip Era (pp. 9-12). IEEE, (2000, April).

[104] H. Arora, R. S. Sherratt, B. Janko and W. Harwin. "Experimental validation of the recovery effect in batteries for wearable sensors and healthcare devices discovering the existence of hidden time constants". The Journal of Engineering, 2017(10), 548-556, (2017).

[105] A. Baumgardt, F. Bachheibl, and D. Gerling, "Utilization of the Battery Recovery Effect in Hybrid and Electric Vehicle Applications", 2014 17th International Con-

ference on Electrical Machines and Systems (ICEMS),Oct. 22-25, Hangzhou, China, 2014.

[106] C. K.Chau, M. H. Wahab, F. Qin, Y. Wang, and Y. Yang, "Battery recovery aware sensor networks", In 2009 7th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (pp. 1-9). IEEE, (2009, June).

[107] C.-K. Chau, F. Qin, S. Sayed, M. H. Wahab, and Y. Yang, "Harnessing battery recovery effect in wireless sensor networks: experiments and analysis," IEEE J. Sel. A. Commun., vol. 28, no. 7, pp. 1222-1232, 2010

[108] L. He, G. Meng, Y. Gu, C. Liu, J. Sun, T. Zhu, L. Yang and K.G. Shin, "Battery-Aware Mobile Data Service", Mobile Computing IEEE Transactions on, vol. 16, pp. 1544-1558, ISSN 1536-1233, 2017.

[109] C. K. Chau, F. Qin, S. Sayed, M. H. Wahab and Y. Yang. "Harnessing battery recovery effect in wireless sensor networks: Experiments and analysis". IEEE Journal on Selected Areas in Communications, 28(7), 1222-1232, (2010).

[110] S. Priya and D. J. Inman, "Energy harvesting technologies", vol. 21: Springer, 2009.

[111] M. K. Stojcev, M. R. Kosanovic, and L. R. Golubovic, "Power management and energy harvesting techniques for wireless sensor nodes," in 2009 9th International Conference on Telecommunication in Modern Satellite, Cable, and Broadcasting Services, pp. 65-72, 2009.

[112] Y. K. Tan and S. K. Panda, "Energy harvesting from hybrid indoor ambient light and thermal energy sources for enhanced performance of wireless sensor nodes," IEEE Transactions on Industrial Electronics, vol. 58, pp. 4424-4435, 2011.

[113] A. Hande, T. Polk, W. Walker, and D. Bhatia, "Indoor solar energy harvesting for sensor network router nodes," Microprocessors and Microsystems, vol. 31, pp. 420-432, 2007.

[114] C. Sauer, M. Stanacevic, G. Cauwenberghs, and N. Thakor, "Power harvesting and telemetry in CMOS for implanted devices," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 52, pp. 2605-2613, 2005.

[115] M. A. Weimer, T. S. Paing, and R. A. Zane, "Remote area wind energy harvesting for low-power autonomous sensors," system, vol. 2, p. 2, 2006.

[116] G. Park, C. R. Farrar, M. D. Todd, W. Hodgkiss and T. Rosing, "Energy Harvesting for Structural Health Monitoring Sensor Networks," Technical Report, Los Alamos National Laboratories, LA, February 2007.

[117] S. Roundy, "Energy Scavenging for Wireless Sensor Nodes with a Focus on Vibration to Electricity Conversion", Ph. D. Dissertation, Dept. of EECS, UC Berkeley, May 2003.

[118] S. Chalasani and J. M. Conrad, "A survey of energy harvesting sources for embedded systems," in Southeastcon, 2008. IEEE, pp. 442-447, 2008.

[119] D. N. Fry, D. E. Holcomb, J. K. Munro, L. C. Oakes, and M. J. Maston, "Compact Portable Electric Power Sources," Report, Oak Ridge National Laboratory, ORNL/TM-13360, 1997.

[120] M. Marzencki, Y. Ammar, and S. Basrour, "Integrated power harvesting system including a MEMS generator and a power management circuit," Sensors and Actuators A: Physical, vol. 145, pp. 363-370, 2008.

[121] $BOLT^{TM}$ INDUSTRIAL Data Sheet, V1.0, in https://www.cornestech.co.jp/images/uploads/file/products/pdf/bolt.pdf, visited on March 1st, 2020.

[122] X. Jiang, J. Polastre, and D. E. Culler, "Perpetual environmentally powered sensor networks," in Proceedings of the Fourth International Symposium on Information Processing in Sensor Networks, IPSN 2005, (UCLA, Los Angeles, California, USA), pp. 463-468, April 25-27, 2005.

[123] V. Raghunathan, A. Kansal, J. Hsu, J. Friedman, and M. B. Srivastava, "Design considerations for solar energy harvesting wireless embedded systems," in Proceedings of IPSN 2005, (UCLA, Los Angeles, California, USA), pp. 457-462, April 25-27, 2005.

[124] I. Mathews, G. Kelly, P. J. King and R. Frizzell, "GaAs solar cells for indoor light harvesting", In 2014 IEEE 40th Photovoltaic Specialist Conference (PVSC) (pp. 0510-0513). IEEE, (2014, June).

[125] G. J. Snyder and A. H. Snyder, "Figure of merit ZT of a thermoelectric device defined from materials properties", Energy and Environmental Science, 10(11), 2280-2283, 2017.

[126] H. Lee, "Thermoelectrics: design and materials", John Wiley and Sons, 2017.

[127] https://thermoelectricsolutions.com/how-thermoelectric-generators-work/, visited on April 1st, 2020.

[128] M. Maas, "Structuration de générateurs thermoélectriques sur échangeur de type radiateur par électrodéposition", (Doctoral dissertation, Université de Lorraine), 2015.

[129] Daedalus: Modular, energy-independent tracking systems, https://www.iis.fraunhofer.de/en/ff/lok/proj/daedalus.html, accessed: 4th April 2017.

[130] P. Blasco, D. Gunduz, M. Dohler, "A learning theoretic approach to energy harvesting communication system optimization", Wireless Communications IEEE Transactions on, vol. 12, no. 4, pp. 1872-1882, (April 2013).

[131] P. Blasco and D. Gunduz, "Multi-access communications with energy harvesting: A multi-armed bandit model and the optimality of the myopic policy," IEEE Journal on Selected Areas in Communications, vol. 33, no. 3, pp. 585-597, (March 2015).

[132] A. Papoulis, "Probability, random variables, and stochastic processes". New York, 1965.

[133] R. Bellman, "Dynamic Programming". Princeton, NJ: Princeton University Press, 1957.

[134] Y. Mansour and S. Singh, "On the complexity of policy iteration", in Proceedings of the 15th International Conference on Uncertainty in AI, Stockholm, SE, pp. 401-408, 1999.

[135] L. Pack Kaelbling, M. L. Littman and A. W. Moore, "Reinforcement Learning: A Survey", Journal of Artificial Intelligence Research 4, pp. 237-285, 1996.

[136] C. Szepesvari, "Reinforcement Learning Algorithm for MDPs", Morganand Claypool Publishers, 2010.

[137] M. Assaouy, O. Zytoune and D. Aboutajdine, "Policy iteration vs Q-Sarsa approach optimization for embedded system communications with energy harvesting", In : 2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). IEEE, p. 1-6, 2017.

[138] M. Assaouy, O. Zytoune and D. Aboutajdine, "DP and RL Approach Optimization for Embedded System Communications with Energy Harvesting", I4CS 2017, CCIS 717, pp. 167-182, 2017.

[139] S. Singh, T. Jaakkola, M. L. Littman, and C. Szepesvari, "Convergence results for single-step on-policy reinforcement-learning algorithms," Machine Learning, vol. 38, no. 3, pp. 287-308, 2000.

[140] M. Corazza and A. Sangalli, "Q-Learning and SARSA: A Comparison between Two Intelligent Stochastic Control Approaches for Financial Trading", University Ca' Foscari of Venice, Dept. of Economics Research Paper Series No. No. 15/WP/2015, (June 10, 2015).

[141] IEEE 802.15.4e Draft Standard: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Std., (March 2010).

[142] M. Assaouy, O. Zytoune and M. Ouadou, "Battery Recovery-Aware Optimization for Embedded System Communications". Wireless Personal Communications, 110(4), 1929-1946, 2020.

[143] A. Galindo-Serrano, L. Giupponi, and M. Dohler, "Cognition and docition in ofdma-based femtocell networks," in IEEE Globecomm, Miami, Florida, USA, pp. 6-10, (Dec. 2010).

[144] E. Evan-Dar and Y. Mansour, "Learning rates for Q-learning," Journal of Machine Learning Research, vol. 5, pp. 1-25, 2003.

[145] P. Auer, T. Jaksch, and R. Ortner, "Near-optimal regret bounds for reinforcement learning," Advances in Neural Information Processing Systems, vol. 21, pp. 89-96, 2009.

[146] N. Salodkar, A. Bhorkar, A. Karandikar, V. S. Borkar, "An on-line learning algorithm for energy efficient delay constrained scheduling over a fading channel," IEEE Journal on Selected Areas in Communications, vol. 26, no. 4, pp. 732-742, Apr. 2008.

[147] W. B. Powell, "Approximate Dynamic Programming: Solving the Curse of Dimensionality", 2nd edition. New York: Wiley, 2011.

[148] M. Assaouy, O. Zytoune and M. Ouadou, "Rapid learning optimization approach for battery recovery-aware embedded system communications", International journal of communication networks and information security, 12, no.3, 2020.

Résumé:

La source d'énergie la plus courante pour les objets connectés est encore principalement basée sur les batteries. Avec une capacité limitée en énergie, ces objets nécessitent continuellement une intervention locale pour remplacer les batteries lorsqu'elles sont vides. Ce qui peut être difficile à réaliser, surtout lorsqu'ils sont placés dans des environnements hostiles et dangereux.

Cette thèse vise à proposer des stratégies de communication permettant d'optimiser les performances des objets connectés utilisant une seule batterie. Les cas de batteries primaires et secondaires ont été pris en compte. Pour les batteries primaires, l'effet de recouvrement a été exploité. Pour les batteries secondaires, la recharge au moyen de récolteurs d'énergie est considérée.

Trois scénarios ont été envisagés en fonction des informations disponibles. Dans le premier cas, il a été supposé que les informations stochastiques sur le système étaient disponibles au début du processus d'optimisation. Dans le second cas, ces informations ont été considérées comme non disponibles et le système supposé totalement inconnu ce qui a nécessité un temps d'apprentissage significatif des politiques optimales. Dans le troisième cas, les informations étaient supposées partiellement connues. Cette connaissance partielle a permis de réduire significativement le temps de calcul pour une performance proche de celle du premier scénario.

***Mots clés**: Internet des objets, Intelligence artificielle, Effet de recouvrement, Récolte d'énergie, Programmation dynamique, Apprentissage par renforcement.*

Abstract:

Most common power source for connected objects still mainly battery-based. With limited power capacity, objects continually require local intervention to replace batteries when empty. Which can be difficult to achieve, especially when placed in hostile and dangerous environments.

This thesis aims at proposing optimal communication strategies allowing the performance optimization of connected objects when using a single battery. Both primary and secondary batteries have been considered. For primaries, we have exploited the battery recovery effect. For secondaries, batteries were assumed to be equipped with energy harvesters.

Three scenarios have been considered depending on the information being available. In the first case, it was assumed that stochastic information was available at the beginning of the optimization process. In the second case, the information was considered unavailable and the system completely unknown which has required a significant amount of time to learn optimal policies. In the third case, it was assumed that the information was partially known. This partial knowledge has significantly reduced the calculation time for a performance close to that of the first scenario.

***Keywords**: Internet of Things, Artificial Intelligence, Recovery effect, Energy harvesting, Dynamic programming, Reinforcement learning.*

Année Universitaire : 2020/2021