ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA
SEDE DI CESENA

Seconda Facoltà di Ingegneria con sede a Cesena
Corso di Laurea Magistrale in Ingegneria Elettronica e delle Telecomunicazioni per lo sviluppo sostenibile

PACKET ERASURE CORRECTING CODES FOR WIRELESS SENSOR NETWORKS: IMPLEMENTATION AND FIELD TRIAL MEASUREMENTS

Tesi in: Sistemi di Telecomunicazioni LM

Presentata da:
MILKA BEKJAROVA

Relatore:
Chiar.mo Prof.
MARCO CHIANI

Correlatori:
Dott. Ing.
ENRICO PAOLINI
Dott. Ing.
MATTEO MAZZOTTI

ANNO ACCADEMICO 2010–2011
SESSIONE III

Keywords

Encoding/Decoding algorithms
Hamming Code
Single Check Parity Code
WSN

Contents

Introduction **xi**

1 Introduction to Wireless Sensor Networks **1**
1.1 Overview of Wireless Sensor Networks 1
1.2 Challenges of Wireless Sensor Networks 2
1.2.1 Characteristic requirements 2
1.2.2 Required mechanisms 3
1.3 Single-Node Architecture 5
1.3.1 Hardware components 5
1.3.2 Operating systems and execution environments 13
1.4 Network applications 17

2 Wireless Sensor Network Standards **21**
2.1 Introduction 21
2.2 IEEE 802.15.4 standard 22
2.2.1 Physical layer (PHY) 23
2.2.2 MAC sublayer 28
2.3 **Data transfer model** 34
2.3.1 Frame structure 37
2.3.2 Beacon frame 38
2.3.3 Data frame 39
2.3.4 Acknowledgment frame 39
2.3.5 MAC command frame 40
2.4 ZigBee higher levels overview 41

3 Packet Correcting Schemes in Wireless Sensor Networks 43
3.1 Packet Erasure Correcting Schemes 43
3.2 Characterization of the transmission chain 46
3.2.1 Characterization of the encoder/decoder 46
3.2.2 Packet correcting coding: Basic concepts 47
3.2.3 Single Check Parity Code 51
3.2.4 Hamming Code 52
3.2.5 Characterization of the transmission channel . . 55
3.3 The simulation of Packet Erasure Ratio (PER) 58
3.3.1 Simulation parameters and flow diagram 59

4 Implementation of Packet Erasure Correcting Codes in Wireless Sensor Networks 65
4.1 Hardware components 65
4.1.1 CC2430 Modules 65
4.1.2 CC2430 Chip 68
4.2 Software environment 69
4.3 TIMAC 71
4.3.1 Nomination of a Personal Area Network (PAN) node in a non-beacon enabled network 72
4.3.2 Non beacon-enabled network Scan and Association 74
4.3.3 Data transactions (Transmission and reception of data packets) 76
4.4 Packet Sniffer 77
4.5 Developed Applications 78
4.6 Implementation of Forward Error Correction (FEC) algorithm using Single Check Parity Code 79
4.7 Implementation of FEC algorithm using Hamming(7,4) code 85
4.8 Implementation of Automatic Repeat and Request (ARQ) protocol 91

5 Results and conclusions 93
5.1 Results of the simulation using a $(10, 9)$ Single Check Parity Code 94

5.2 Results of the simulation using Hamming (7,4) Code . 95
5.3 Experimental Activity 97
5.3.1 Using the FEC technique for recovery of the packet erasures 98
5.3.2 Using the ARQ technique for the recovery of the packet erasures 99
5.3.3 Results 100
5.4 Conclusions 104
5.5 Problems observed 104
5.6 Future work 105

Bibliography **107**

Acknowledgments **115**

Acronyms

ACK	Acknowledgment Frame
ARQ	Automatic Repeat and Request
API	Application Programming Interface
BEC	Binary Erasure Channel
BER	Bit Error Rate
CAP	Contention Access Period
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DMC	Discrete Memoryless Channel
FEC	Forward Error Correction
FFD	Full Function Device
GTS	Guaranteed Time Slots
HAL	Hardware Abstraction Layer
HARQ	Hybrid Automatic Repeat and Request
MAC	Medium Access Channel
OSAL	Operating System Abstraction Layer
PAN	Personal Area Network
PCC	Packet Correcting Codes

PDR	Packet Delivery Ratio
PEC	Packet Erasure Channel
PER	Packet Erasure Ratio
PHY	Physical Layer
PLR	Packet Loss Ratio
QoS	Quality of Service
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Introduction

The considerable mobile services sector growth around the world was certainly the major phenomenon of the 1990s in the telecommunications field. There has been rapid development and advancement in the communication and sensor technologies that results in the growth of a new, attractive and challenging research area - the wireless sensor network. This thesis regards the *Wireless Sensor Network (WSN)*, as one of the most important technologies for the twenty-first century and the implementation of different packet correcting erasure codes to cope with the "bursty" nature of the transmission channel and the possibility of packet losses during the transmission. The limited battery capacity of each sensor node makes the minimization of the power consumption one of the primary concerns in WSN. Considering also the fact that in each sensor node the communication is considerably more expensive than computation, this motivates the core idea to invest computation within the network whenever possible to safe on communication costs. The goal of the research was to evaluate a parameters (for example the *Packet Erasure Ratio* (PER)) that permit to verify the functionality and the behavior of the created network, validate the theoretical expectations and evaluate the convenience of introducing the recovery packet techniques using different types of packet erasure codes in different types of networks. Thus, considering all the constrains of energy consumption in WSN, the topic of this thesis is to try to minimize it by introducing encoding/decoding algorithms in the transmission chain in order to prevent the retransmission of the erased packets through the *Packet Erasure Channel* and save the energy used for each retransmitted packet. In this way it is possible extend the lifetime of entire network.

The main characteristics of WSNs, the challenges, the architecture

of a single wireless node and the description of different topologies of network are presented in Chapter 1. Chapter 2 deals with the standardization of wireless sensor networks, which proceeds along two main directives: the IEEE 802.15.4 standard [8] and ZigBee [16]. The standard defines the protocol and interconnection of devices via radio communication in a personal area network. The Physical Layer (PHY) and Medium Access Channel (MAC) sublayer proposed by the IEEE 802.15.4 standard are reviewed and illustrated.
The "bursty" and noisy nature of the transmission channel has originate the idea of the introduction of packet recovery schemes in this type of networks and the possibility to prevent the possible packet erasures and losses through the channel. In Chapter 3 are presented the packet correcting schemes mainly used in wireless systems that can be divided into three main categories: *Automatic Repeat and Request*, that provides in case of packet transmission erasures the solution to retransmit the entire packet again; *Forward Error Correction*, that provides introducing redundant information and by applying decoding algorithm that uses these redundant packets it is possible the recovery of the packet losses, instead of their retransmission and *Hybrid Automatic Repeat and Request*. The ARQ approach is simple to use but the disadvantage of using is the additional retransmission energy cost. Energy constrained transmission of WSN makes the alternative FEC approach a popular technique.
So, in this thesis we tried to implement a FEC algorithms using two different types of codes (Single Check Parity Code and Hamming Code) better discussed at Chapter 3, trying the recovery of the packet losses at the receiver node by introducing an encoding/decoding techniques in the transmission chain. The only issue of the decoding is that not every time the results are successfully received packets. If the number or erased packets is greater than the number of packets that the decoder can successfully recover (which depends of the code rate and the level of redundancy packets introduced), the decoding process fails and the erased packets can't be recover. Also, quantity of redundancy packets (overhead) introduced has to be taken into account. All this considerations, the realization and implementation of the encoding/decoding algorithms are better addressed in Chapter 4.
In order to confirm the theoretical expectations, two real-time applica-

tions were implemented using the development kit (Chipcon CC2430EM and CC2430EB) from Texas Instruments. The description of the instruments used is presented at Chapter 4. The experimentation was made in different conditions changing the code used, the topology of the network, the distance between the sensor nodes and the transmission power. Chapter 5 reports the results and the conclusions obtained during the different tests made.

Chapter 1

Introduction to Wireless Sensor Networks

1.1 Overview of Wireless Sensor Networks

WSN have been widely considered as one of the most important technologies for the twenty-first century. Enabled by recent advances in microelettronicmechanical systems (MEMS) and wireless communication technologies, tiny, cheap, and smart sensor deployed in a physical area and networked through wireless links and the Internet, provide unprecedented opportunities for a variety of civilian and military applications, for example, environmental monitoring, battle field surveillance and industry process control. The WSNs have unique characteristics: higher unreliability of sensor nodes, denser level of node deployment, and severe energy, computation and storage constraints which presents many new challenges in their development and applications. In the past decade, WSNs have received attention from both academia and industry all over the world. In order to explore various design and application issues, a large amount of research activities have bee carried out and significant advances have been made at the deployment of WSNs. In the near future WSNs will be widely used in various civilian and military fields, and will revolutionize the way we live, work and interact with the physical world.

1.2 Challenges of Wireless Sensor Networks

1.2.1 Characteristic requirements

The following characteristics are shared among most of the applications that make use of WSN:

Type of service. A WSN, is expected to provide meaningful information and/or actions about a given task. Additionally, concepts like *scoping* of interactions to specific geographic regions or to time intervals will become important. Hence, new paradigms of using such a network are required, along with new interfaces and new ways of thinking about the service of a network.

Quality of Service. Closely related to the type of a network's service is the quality of that service. In some cases, only occasional delivery of a packet can be more than enough; in other cases, very high reliability requirements exist. In yet other cases, delay is important when actuators are to be controlled in a real-time fashion by the sensor network. The packet delivery ratio is an insufficient matric; what is relevant is the amount and quality od information that can be extracted at given sinks about the observed objects or area.

Fault tolerance. Since nodes may run out of energy or might be damaged, or since the wireless communication between two nodes can be permanently interrupted, it is important that the WSN is able to tolerate such faults. To tolerate node failure, redundant deployment is necessary, using more nodes than would be strictly necessary if all nodes functioned correctly.

Lifetime. In many scenarios, the wireless nodes will have to rely on a limited supply of energy (using batteries). Replacing these energy sources in the field is usually not practicable, and simultaneously, a WSN must operate as long as possible. Hence, the lifetime of a WSN becomes very important figure of merit. Evidently, an energy-efficient way becomes very important for

the WSN. As an alternative to energy supplies, a limited power source (via power sources like solar cells, for example) might also be available on a sensor node. Typically, these sources are not powerful enough to ensure continuous operation but can provide some recharging of batteries. The lifetime of a network also has direct trade-offs against quality of service: investing more energy can increase quality but decrease lifetime. Concepts to harmonize these trade-offs are required.

Scalability. The employed architectures and protocols must be able to scale, since a WSN might include a large number of nodes.

Wide range of densities. In a WSN, the number of nodes per unit area-the *density* of the network - can vary considerably. Even within a given application, density can vary over time and space because fail or move.

Programmability. These nodes should be programmable and not only will be necessary for the nodes to process information but also, their programming must be changeable during operation when new tasks become important. A fixed way of information processing is insufficient.

Maintainability. Considering the fact that both the environment of a WSN and the WSN itself change (depleted batteries, failing nodes), the system has to adapt. In this case, the network has to maintain itself; it could also be able to interact with external maintenance mechanisms to ensure its extended operation as a required quality [3].

1.2.2 Required mechanisms

To realize the requirements described above, have to be developed innovative mechanisms for a communication network, as well as new protocol concepts and architectures. A particular challenge here is the need to find mechanisms that are sufficiently specific for a given application to support the specific lifetime, quality od service, and maintainability requirements [4].
Some of the mechanisms that make part of WSNs are:

- ***Multihop wireless communication.*** In a wireless system a direct communication between a sender and a receiver is faced with limitations. In particular, communication over long distance is only possible using high transmission power. The use of intermediate nodes as relays can reduce the total required power. Hence, for many forms of WSNs, so-called multihop communication will be a necessary ingredient.

- ***Energy-efficient operation.*** To support long lifetimes, energy-efficient operation is a key technique. Options to look into include energy-efficient data transport between two nodes (measured in J/bit) or, more importantly, the energy-efficient determination of a requested information. Also, nonhomogeneous energy consumption - the forming of "hotspots" - is an issue.

- ***Auto-configuration.*** Independent of external configuration a WSN will have to configure most of its operational parameters autonomously, - the sheer number of nodes and simplified deployment will require that capability in most applications. As an example, nodes should be able to determinate their geographical positions only using other nodes of the network - so-called "self-location". Also, the network should be able to tolerate failing nodes (because of a depleted battery, for example) or to integrate new nodes (because of incremental deployment after failure, for example).

- ***Collaboration and in-network processing.*** In some applications, several sensors have to collaborate to detect an event and only the joint data of many sensors provides enough information. A single sensor is not able to decide whether an event has happened. To solve such tasks efficiently, readings from individual sensors can be aggregated as they propagate through the network, reducing the amount of the data to be transmitted and hence improving the energy efficiency.

- ***Data centric.*** Traditional communication networks are typically data-centric networks around the transfer of data between two specific devices, each equipped with (at least) one network

address - the operation of such networks is thus address-centric. In a WSN, where nodes are typically deployed redundantly to protect against node failures or to compensate for the low quality of a single node's actual sensing equipment, the identity of the particular node supplying data becomes irrelevant in some cases. Hence, switching from an address-centric paradigm to a data-centric paradigm in designing architecture and communication protocols is promising.

- ***Locality.*** The principle of locality will have to be embraced extensively to ensure, in particular, scalability. Nodes, which are very limited in resources like memory, should attempt to limit the state that they accumulate during protocol processing to only information about their direct neighbors. The hope is that this will allow the network to scale to large numbers of nodes without having to rely on powerful processing at each single node. How to combine the locality principle with efficient protocol design is still an open research topic.

1.3 Single-Node Architecture

The nodes that make part od a WSNs have to meet the requirements that come from the specific requirements of a given application: they have to be equipped with the right sensors, the necessary computation and memory resources, they might have to be small, cheap, and energy efficient, and they need adequate communication facilities. These hardware components and their composition into a functioning node are described in Section 1.3.1; In addition to the hardware of sensor nodes, the operating system and programming model is an important consideration. Section 1.3.2 describes the tasks of such an operating system along with some examples as well as suitable programming interfaces.

1.3.1 Hardware components

When choosing the hardware components for a wireless sensor node, evidently the application's requirements play a decisive factor with

regard mostly to size, costs, and energy consumption of the nodes-communication facilities as such are often considered to be acceptable quality, but the trade-offs between features and costs is crucial. In some extreme visions, the nodes are sometimes claimed to have to be reduced to the size of grains of dust. In more realistic applications, the mere size of a node is not so important; rather, convenience and simple power supply are more important [5].

A basic sensor node comprises five main components (Figure 1.1):

Controller. A controller to process all the relevant data, capable of executing arbitrary code.

Memory. Some memory to store programs and intermediate data; usually, different types of memory are used for programs and data.

Sensor and actuators. The actual interface to the physical world: devices that can observe or control physical parameters of the environment.

Communication. Tuning nodes into a network requires a device for sensing and receiving information over a wireless channel.

Power supply. As usually no tethered power is available, some form of batteries are necessary to provide energy. Sometimes, some form of recharging by obtaining energy from the environment is available as well (e.g. solar cells).

Each of this components has to operate balancing the trade-off between as small an energy consumption as possible on the one hand and the need to fulfill their tasks on the other hand. For example, both the communication device and the controller should be turned off as long as possible. To wake up again, the controller could, for example, use a preprogrammed timer to be reactivated after some time. Alternatively, the sensors could be programmed to raise an interrupt if a given event occurs, a temperature value exceeds a given threshold or the communication device detects an incoming transmission.
Supporting such alert functions requires appropriate interconnection between individual components. Moreover, both control and data information have to be exchanged along these interconnections. This

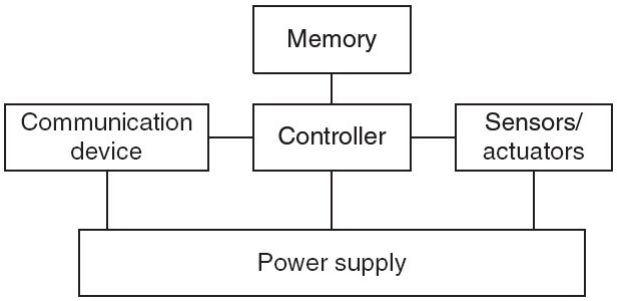

Figure 1.1: *Overview of main sensor node hardware components* [6].

interconnection can be very simple - for example, a sensor could simply report an analog value to the controller or it could be endowed with some intelligence of its own, preprocessing sensor data and only waking up the main controller if an actual event has been detected. Such preprocessing can be highly customized to the specific sensor yet remain simple enough to run continuously, resulting in improved energy efficiency [7].

Controller

The controller is the core of a wireless sensor node. It collects data from the sensors, processes this data, decides where to send it and receives data from other sensor nodes. It has to execute various programs, it is the Central Processing Unit (CPU) of the node. Such a variety of processing tasks can be performed on various controller architectures, representing trade-offs between performance, flexibility, energy efficiency, and costs.
One solution is to use general-purpose processors, like those known from desktop computers. These processors are highly overpowered, and their energy consumption is excessive. But simpler processors do exist, specifically used in embedded systems. These processors are commonly referred as **microcontrollers.** Some of the key characteristics why these microcontrollers are particulary suited to embedded systems are their flexibility in connecting with other devices (like sensors) and their typically low power consumption. In addition, they are very flexible and freely programmable. Microcontrollers are also

suitable for WSNs since they commonly have the possibility to reduce their power consumption by going into **sleep states** where only parts of the controller are active.
Microcontrollers that are used in several wireless sensor node prototypes include the Atmel processor or Texas Instrument's MSP 430. In older prototypes, the Intel StrongArm processors have also been used. Nonetheless, as the principal properties of these processors and controllers are quite similar, conclusions from these earlier research results still hold to a large degree.

Memory

The memory component is fairly straightforward. The Random Access Memory (RAM) is used to store packets from other nodes, intermediate sensor readings, and so on. While RAM is fast, its main disadvantage is that it loses its content if power supply is interrupted. Program code can be stored in Read-Only-Memory (ROM) or, more typically, in Electrically Erasable Programmable Read-Only-Memory (EEPROM) or flash memory.
Correctly dimensioning memory sizes, especially RAM, can be crucial with respect to manufacturing costs and power consumption.

Communication device

The communication device is used to exchange data between individual nodes. The usage of Radio Frequency (RF)-based communication best fits the requirements of most WSN applications: It provides acceptable error rates at reasonable energy expenditure, relatively long range and high data rates, and does not require line of sight between sender and receiver.
For a RF-based system, the carrier frequency has to be carefully chosen. WSNs typically use communication frequencies between about 433 MHz and 2.4 GHz.

Transceivers

For actual communication, both a transmitter and a receiver are required in a sensor node. The essential task is to convert a bit stream

coming from a microcontroller (or a sequence of bytes or frames) and convert them to and from radio waves. A device that combines these two tasks in a single entity is called **transceiver.**

Transceiver structure

A fairly common structure of transceivers is into the Radio Frequency (RF) front end and the baseband part:

- the **radio frequency front end** performs analog signal processing in the actual radio frequency band, whereas
- the **baseband processor** performs all signal processing using a sensor node's processor or other digital circuitry.

Between these two parts, a frequency conversion takes place, either directly or via one or several Intermediate Frequencies (IFs). The **RF front end** performs analog signal processing in the actual radio frequency band, for example in the 2.4 GHz Industrial, Scientific, and Medical (ISM) band; it is the first stage of the interface between the electromagnetic waves and the digital signal processing of the further transceiver stages[9].

Transceiver operational states

Many transceivers can distinguish four operational states[10]:

Transmit. In the transmit state, the transmit part of the transceiver is active and the antenna radiates energy.

Receive. In the receive state the receive part is active.

Idle. A transceiver that is ready to receive but is not currently receiving anything is said to be in an **idle state**. In this idle state, many parts of the receive circuitry are active, and others can be switched off.

Sleep. In the **sleep state**, significant parts od the transceiver are switched off. There are transceivers offering several different sleep states. These sleep states differ in the amount of circuitry

switched off and in the associated **recovery times** and **startup energy**[11].

The sensor node's protocol stack and operating software must decide into which state the transceiver is switched, according to the current and anticipated communications needs.

Some examples of radio transceivers

To complete this discussion of possible communication devices, a few examples of standard radio transceivers that are commonly used in various WSN prototype nodes should be briefly described.

RFM TR1000 family

The TR1000 family of radio transceivers from PF Monolithics[1] is available for the 916 MHz and 868 MHz frequency range. It works in a 400 kHz wide band centered at, for example, 916.50 MHz. It is intended for short-range radio communication with up to 115.2 kbps. The modulation is either on-off keying (at a maximum rate of 30 kbps) or ASK; it also provides a dynamically tunable output power. The transceiver offers received signal strength information. It is attractive because of its low-power consumption in both send and receive modes and especially in sleep mode.

Chipcon CC1000 and CC2420 family

Chipcon[2] offers a wide range of transceivers that are appealing for use in WSN hardware. To name but two examples: the CC1000 operates in a wider frequency range, between 300 and 1000 MHz, programmable in steps of 250 Hz. It uses FSK as modulation, provides RSSI, and has programmable output power. An interesting feature is the possibility to compensate for crystal temperature drift. It should also be possible to use it in frequency hopping protocols. Details can be found in the data sheet[12].

[1] http://www.rfm.com/
[2] http://www.chipcon.com/

The CC2420[13] is more complicated device. It implements the physical layer as prescribed by the IEEE 802.15.4 standard with the required support for this standard's MAC protocol. In fact, the company claims that this is the first commercially available single-chip transceiver for IEEE 802.15.4. As a consequence of implementing this standard, the transceiver operates in the 2.4 GHz band and features the required DSSS modem, resulting in a data rate of 250 kbps. It achieves this at still relatively low-power consumption, although not quite on par with the simpler transceivers described so far.

Sensors and actuators

Without the actual sensors and actuators, a wireless sensor network would be beside the point entirely.
Sensors can be roughly categorized into three categories:

1. **Passive, omnidirectional sensors.** These sensors can measure a physical quantity at the point of the sensor node without actually manipulating the environment. Moreover, some of these sensors actually are self-powered in the sense that they obtain the energy they need from the environment - energy is only needed to amplify their analog signal. There is no notion of "direction" involved in these measurements. Typical examples for such sensors include thermometer, light sensors, vibration, microphones, humidity, mechanical stress or tension in materials, chemical sensors sensitive for given substances, smoke detectors, air pressure, and so on.

2. **Passive, narrow-beam sensors.** These sensors are passive as well, but have a well-defined notion of direction of measurement. A typical example is a camera, which can "take measurements" in a given direction, but has to be rotated if need be.

3. **Active sensors.** This last group of sensors actively probes the environment, for example, a sonar radar or some types of seismic sensors, which generate shock waves by small explosions. These are quite specific-triggering an explosion is certainly not a lightly undertaken action, and require quite special attention.

In practice, sensors from all of these types are available in many different forms with many individual peculiarities. Obvious trade-offs include accuracy, dependability, energy consumption, cost, size, and so on - all this would make a detailed discussion of individual sensors quite ineffective.

Actuators

Actuators are just about as diverse as sensors, yet for the purposes of designing a WSN, they are a bit simpler to take account of: In principle, all that a sensor node can do is to open or close a switch or a relay or to set a value in some way. Hether this controls a motor, a light bulb, or some other physical object is not really of concern to the way communication protocols are designed.
In a real network, however, care has to be taken to properly account for the idiosyncrasies of different actuators. Also, it is good design practice in most embedded system applications to pair any actuator with a controlling sensor.

Energy consumption of sensor nodes

Energy supply for a sensor node is at a premium: batteries have small capacity and recharging by energy scavenging is complicated and volatile. Hence, the energy consumption of a sensor node must be tightly controlled.
One important contribution to reduce power consumption of these components comes from chip-level and lower technologies: Designing low-power chips is the best starting point for an energy-efficient sensor node. But this is only the half of the picture, as any advantages gained by such designs can easily be squandered when the components are improperly operated.
The crucial observation for proper operation is that most of the time a wireless sensor node has nothing to do. Hence, it is best to turn it off. Naturally, it should be able to wake up again, on the basis of external stimuli or on the basis of time. Therefore, completely turning off a node is not possible, but rather, its operational state can be adapted to the tasks at hand. Introducing and using multiple states of operation with reduced consumption in return for reduced functionality is

the core technique for energy-efficient wireless sensor node.
These modes can be introduced for all components of a sensor node, in particular, for controller, radio front end, memory, and sensors. Different models usually support different numbers of such sleep states with different characteristics. For a controller, typical states are "active", "idle", and "sleep"; a radio modem could turn transmitter, receiver, or both on or off;sensors and memory could also be turned on or off.

The main consumers of energy are the radio front ends, the controller, to some degree the memory, and, depending on the type, the sensors. Looking at a confront of energy consumption numbers for different types of microcontrollers and radio transceivers, an evident question to ask is which is the best way to invest the precious energy resources of a sensor node: Is it better to send data or to compute? What is the relation in energy consumption between sending data and computing?
Normally, when in transmit mode the transceiver drains much more current from the battery than the microprocessor in active state, or the sensors and the memory chip. The ratio between the energy needed for transmitting and for processing a bit of information is usually assumed to be much larger than one (more then one hundred or one thousand in most commercial platforms). For this reason, the communication protocols need to be designed according to energy-efficient paradigms, while processing tasks are not, usually [16].

It is clear that communication is a considerably more expansive undertaking than computation, (Figure 1.2). This basic observation motivates a number of approaches and design decisions for the networking architecture of wireless sensor networks. The core idea is to invest into computation within the network whenever possible to safe on communication costs, leading to the notion of in-network processing and aggregation. These ideas will be discussed in Chapter 3.

1.3.2 Operating systems and execution environments

An operating system or an execution environment - for WSNs should support the specific needs of these systems. In particular, the need for energy-efficient execution requires support for energy management.

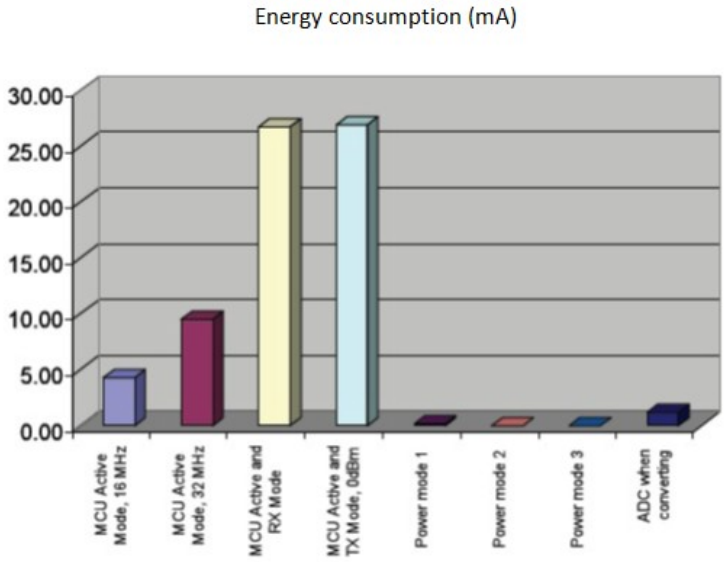

Figure 1.2: *Energy consumption in a single node*[16]

Also, external components - sensors, the radio modem, or timers - should be handled easily and efficiently, in particular, information that becomes available asynchronously (at any arbitrary point in time) must be handled.
All this requires an appropriate programming model, a clear way to structure a protocol stack, and explicit support for energy management - without imposing too heavy a burden on scarce system resources like memory or execution time.

Programming paradigms and application programming interfaces

Concurrent Programming

One of the first questions for a programming paradigm is how to support concurrency. Such support for concurrent execution is crucial for WSN nodes, as they have to handle data communing from arbitrary sources - for example, multiple sensors or the radio transceivers - at arbitrary points in time. A system could poll a sensor to check whether a packet is available, and process the data right away, then

poll the transceiver to check whether a packet is available, and then immediately process the packet, and so on (Figure 1.3). Such a simple sequential model would run the risk of missing data while a packet is processed or missing a packet when sensor information is processed. This risk is particulary large if the processing of sensor data or incoming packets takes substantial amounts of time, which can easily be the case. Hence, a simple sequential programming model is clearly insufficient.

Event-based programming

Most modern, general-purpose operating systems support concurrent (seemingly parallel) execution of multiple processes on a single CPU. Hence, such a process-based approach would be a first candidate to support concurrency in a sensor node as well; it is illustrated in (b) of Figure 1.3. The idea is to embrace the reactive nature of a WSN node and integrate it into the design of the operating system. The system essentially waits for any event to happen, where an event typically can be the availability of data from sensor, the arrival of a packet, or the expiration of a timer. Such an event is than handled by a short sequence of instructions that only stores the fact that this event has occurred and stores the necessary information. The actual processing of this information is not done in these event handler routines, but separately, decoupled from the actual appearance of events. This **event-based programming** [17] model is sketched in Figure 1.4.

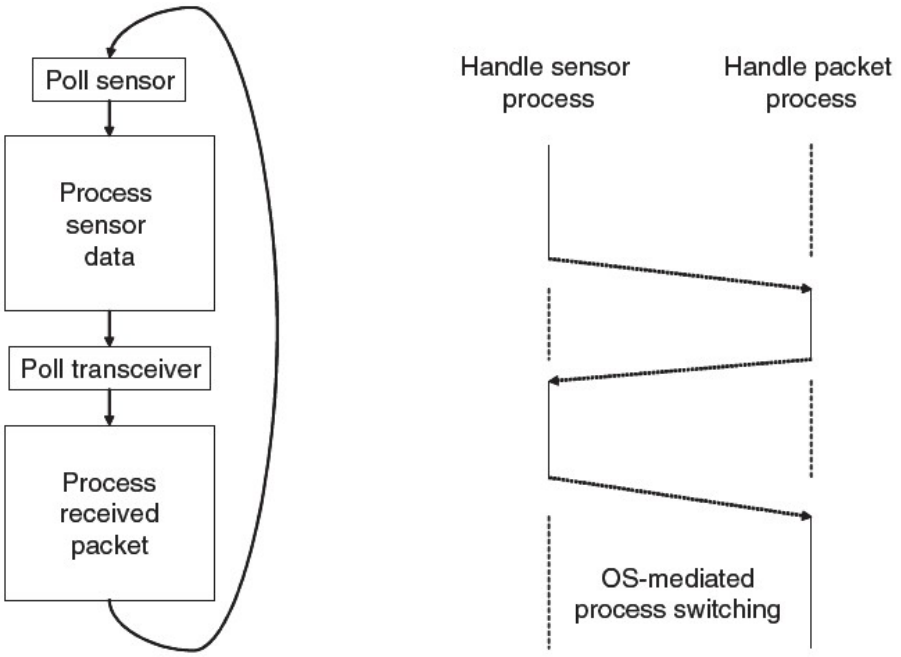

Figure 1.3: *Two programming models for WSN operating systems:purely sequential execution (a) and process-based execution (b)*[6]

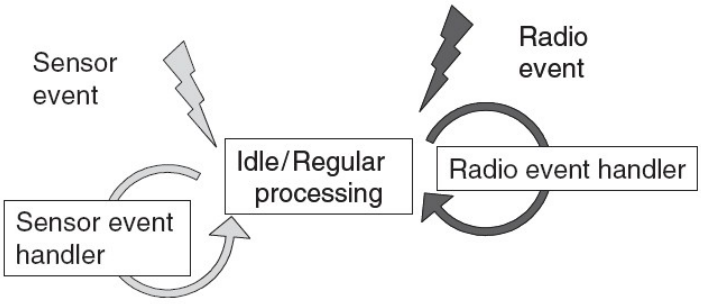

Figure 1.4: *Event-based programming model*[6]

1.4 Network applications

Sensors can be used to detect or monitor a variety of physical parameters or conditions [5], for example:

- Light
- Sound
- Humility
- Pressure
- Temperature
- Soil composition
- Air or water quality
- Attributes of an object such as size, weight, position, and direction.

Wireless sensors have significant advantages over the conventional wired sensors. They can not only reduce the cost and delay in deployment, but also be applied to any environment, especially those in which conventional wired sensor networks are impossible to be deployed, for example, inhospitable terrains, battlefields, outer space, or deep oceans. WSNs were originally motivated by military applications, which range from large-scale acoustic surveillance systems for

ocean surveillance to small networks of unattended ground sensors for ground target detection. However, the availability of low-cost sensors and wireless communication has promised the development of a wide range of applications in both civilian and military fields. This section introduces a few examples of sensor network applications.

Environmental Monitoring. Environmental monitoring is one of the earliest applications of sensor networks. In environmental monitoring, sensors are used to monitor a variety of environmental parameters or conditions.

- *Habitat Monitoring.* Sensors can be used to monitor the conditions of wild animals or plants in wild habitats, as well as the environmental parameters of the habitat, for example humidity, pressure, temperature, and radiation.
- *Air or Water Quality Monitoring.* Sensors can be deployed on the ground or under water to monitor air or water quality. For example, water quality monitoring can be used in the hydrochemistry field. Air quality monitoring can be used for air pollution control.
- *Hazard Monitoring.* Sensors can be used to monitor biological or chemical hazards in locations, for example, a chemical plant or a battlefield.
- *Disaster Monitoring.* Sensors can be densely deployed in an intended region to detect natural or non-natural disasters. For example, sensors can be scattered in forests or rivers to detect fires or floods. Seismic sensors can be instrumented in a building to detect the direction and magnitude of a quake and provide an assessment of the building safety.

Military Applications. WSNs are becoming an integral part of military command, control, communication, and intelligence systems. Wireless sensors can be rapidly deployed in a battlefield or hostile region without any infrastructure. Due to ease of deployment, self-configurability, untended operation, and fault tolerance, sensor networks will play more important rules in future military systems.

Health Care Applications. WSNs can be used to monitor and track elders and patients for health care purposes, which can significantly relieve the severe shortage of health care personnel and reduce the health care expenditures in the current health care systems.

Facility management. WSNs also have a wide range of possible applications also in this area. Simple examples include keyless entry applications where people wear badges that allow a WSN to check which person is allowed to enter which areas of a larger company site. This example can be extended to the detection of intruders for example of vehicle's position and alert security personnel.

Machine surveillance and preventive maintenance. One idea is to fix sensor nodes to difficult-to-reach areas of machinery where they can detect vibration patterns that indicate the need for maintenance. Examples for such machinery could be robotics or the axles of trains. The main advantage of WSNs here is the cablefree operation, avoiding a maintenance problem in itself and allowing a cheap, often retrofitted installation of such sensors.

Precision agriculture. Applying WSN to agriculture allows precise irrigation and fertilizing by placing humidity/soil composition sensors into the fields. Also, livestock breeding can benefit from attaching a sensor to each animal, which controls the health status (by checking body temperature, step counting, or similar means) and raises alarms if given thresholds are exceeded.

Chapter 2

Wireless Sensor Network Standards

2.1 Introduction

Robust, reliable wireless sensor networks stand to benefit a number of industries, and as a result much effort has been expended in recent years to develop design standards for WSNs. The standardization proceeds along two main directives: the IEEE 802.15.4 standard [8] and ZigBee [16]. These two standards specify different subsets of layers: IEEE 802.15.4 defines the PHY and MAC Layer, and ZigBee defines the network and application layers, as shown in Figure 2.1. The two protocol stacks can be combined to support low data rate and long-lasting applications on battery-powered wireless devices. Application fields of these standards include sensors, interactive toys, smart badges, remote controls, and home automation.

The first release of IEEE 802.15.4 was delivered in 2003 and it is freely distributed. This standard was revisited in 2006. The ZigBee protocol stack was proposed at the end of 2004 by the ZigBee alliance, an association of companies working together to develop standards (and products) for reliable, cost-effective, low-power wireless networking. The first release of ZigBee has been revised at the end of 2006. The 2006 version introduces extensions relating the standardization of application profiles and some minor improvements to the network and application layers. This Chapter will focus on analyzing the sub-

layers proposed by IEEE 802.15.4 and the main functionalities shared by the two releases of ZigBee. It is organized in two parts: the first part, given in Section 2.2, introduces the PHY and MAC layers proposed by IEEE 802.15.4 and the second part presents the network and application layers of ZigBee, which is given in Section 2.3.

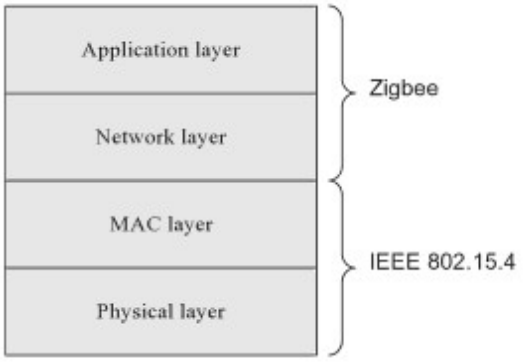

Figure 2.1: *The protocol stack of the IEEE 802.15.4 and ZigBee Standards*[18].

2.2 IEEE 802.15.4 standard

The IEEE 802.15.4 standard [8] specifies the PHY and MAC layers for low-rate Wireless Personal Area Network (WPAN). Its protocol stack is simple, and does not require any infrastructure, which is suitable for short-range communications (typically within a range of 100m). For these reasons, it features ease of installation low cost and a reasonable battery life of the devices. The IEEE 802.15.4 architecture is defined in terms of a number of blocks in order to simplify the standard and offers services to the higher layers. The layout of the blocks is based on the open systems interconnection (OSI) seven-layer model (ISO/IEC 7498-1:1994).

An low-rate WPAN device comprises a PHY, which contains the radio frequency (RF) transceiver along with its low-level control mechanism, and a MAC sublayer that provides access to the physical channel

for all types of transfer. Figure 2.2 shows these blocks in a graphical representation.

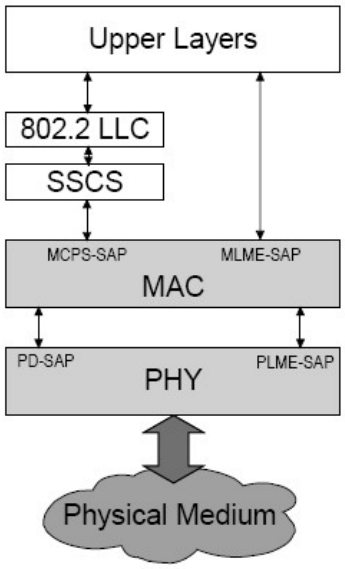

Figure 2.2: *WPAN device architecture*[8]

The upper layers, shown in Figure 2.2, consist of a network layer, which provides network configuration, manipulation, and message routing, and an application layer, which provides the intended function of the device. An IEEE 802.2 Type 1 logical link control (LCC) can access the MAC sublayer through the service-specific convergence sublayer (SSCS).

2.2.1 Physical layer (PHY)

The physical layer of the IEEE 802.15.4 standard has been designed to coexist with other IEEE standards for wireless networks, for example, IEEE 802.11 and IEEE 802.15.1 (Bluetooth). The PHY provides two services: the PHY data service and the PHY management service interfacing to the physical layer management entity (PLME) service access point (SAP). The PHY data service enables the transmission

and reception of PHY protocol data units (PPDUs) across the physical radio channel. The radio operates in one of the following three license-free bands:

- 868-868.6 MHz (e.g., Europe) with a data rate of 20 kbps.

- 902-928 MHz (e.g., North America) with a data rate of 40 kbps.

- 2400-2483.5 MHz (worldwide) with a data rate of 250 kbps.

General requirements and definitions

The PHY is responsible for the following tasks:

- Activation and deactivation of the radio transceiver;
- Energy detection (ED) within the current channel;
- Link quality indicator (LQI) for received packets;
- Clear channel assessment (CCA) for Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA);
- Channel frequency selection;
- Data transmission and reception.

The standard specifies the following four PHYs:

An 868/915 MHz direct sequence spread spectrum (DSSS) PHY employing binary phase-shift keying (BPSK) modulation.

An 868/915 MHz DSSS PHY employing offset quadrature phase-shift keying (O-QPSK) modulation.

An 868/915 MHz parallel sequence spectrum (PSSS) PHT employing BPSK and amplitude shift keying (ASK) modulation.

A 2450 MHz DSSS PHY employing O-QPSK modulation.

In addition to the 868/915 MHz BPSK PHY, which was originally specified in the 2003 edition of this standard, two optional high-data-rate PHYs are specified for the 868/915 MHz bands, offering a tradeoff between complexity and data rate. Both optional PHYs offer a data rate much higher than that of the 868/915 MHz BPSK PHY, which provides for 20 kb/s in the 868 MHz band and 40 kb/s in the 915 MHz band. The ASK[1] PHY offers data rates of 250 kb/s in both the 868 MHz and 915 MHz bands, which is equal to that of the 2.4 GHz band PHY. The O-QPSK PHY, which offers a signaling scheme identical to that of the 2.4 GHz band PHY, offers a data rate in the 915 MHz band equal to that of the 2.4 GHz band PHY and a data rate of 100 kb/s in the 868 MHz band.

Operating frequency range

A compliant device shall operate in one or several frequency bands using the modulation and spreading formats summarized in Figure 2.3. Devices shall start in the mode (PHY) they are instructed to do

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate(ksymbol/s)	Symbols
868/915	868-868.6	300	BPSK	20	20	Binary
	902-928	600	BPSK	40	40	Binary
868/915 (optional)	868-868.6	400	ASK	250	250	20-bit PSSS
	902-928	1600	ASK	250	250	5-bit PSSS
868/915 (optional)	868-868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902-928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400-2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

Figure 2.3: *Frequency bands and data rates*[8]

so. If the device is capable of operating in the 868/915 MHz bands using one of the optional PHYs, it shall be able to switch dynamically

[1]The 868/915 MHz band ASK PHYs use BPSK moulation for the SHR and ASK modulation for the remainder of the PPDU.

between the optional 868/915 MHz band PHY and the mandatory 868/915 MHz BPSK PHYs when instructed to do so. This standard is intended to confirm with established regulations in Europe, Japan, Canada, and the United States. Devices conforming to this standard shall also comply with specific regional legislation.

Channel assignments

The introduction of the "868/915 MHz band (optional) amplitude shift keying (ASK) PHY specifications" and "868/915 MHz band (optional) O-QPSK PHY specifications" results in the total number of channel assignments exceeding the channel numbering capability of 32 channel numbers that was defined in the 2003 edition of this standard. To support the growing number of channels, channel assignments shall be defined through a combination of channel numbers and channel pages.

A total of 27 channels numbered from 0 to 26 are available across the three frequency bands. Sixteen channels are available in the 2450 MHz band, 10 in the 915 MHz band, and 1 in the 868 MHz band.

PPDU format

For convenience, the PPDU packet structure is presented so that the leftmost field as written in this standard shall be transmitted or received first. All multiple octets fields shall be transmitted or received least significant octet first and each octet shall be transmitted or received least significant bit (LSB) first. The same transmission order should apply to data fields transferred between the PHY and MAC sublayer.
Each PPDU packet consists of the following basic components:

- A synchronization header (SHR), which allows a receiving device to synchronize and lock onto the bit stream
- A PHY header (PHR), which contains length information
- A variable length payload, which carries the MAC sublayer frame

The PPDU packet structure shall be formatted as illustrated in Figure 2.4.

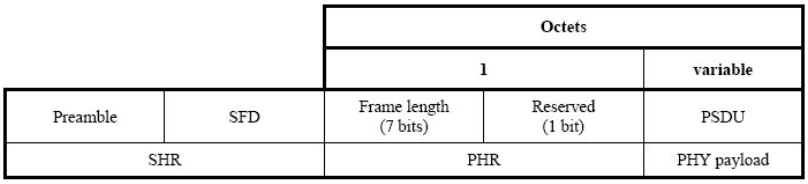

Figure 2.4: *Format of the PPDU[8]*

Preamble field. The preamble field is used by the transceiver to obtain the chip and symbol synchronization with an incoming message.

SFD field. The SFD is a field indicating the end of he SHR and the start of the packet data.

Frame Length field. The Frame Length field is 7 bits in length and specifies the total number of octets contained in the PSDU (i.e., PHY payload). It is a value between 0 and *MaxPHYPacketSize*.

PSDU field. The PSDU field has a variable length and carries the data of the PHY packet.

2.4 GHz PHY specifications

The data rate of the IEEE 802.15.4 (2450 MHz) PHY shall be 250 kb/s. It employs a 16-ary quasi-orthogonal modulation technique. During each data symbol period, four information bits are used to select one of the 16 nearly orthogonal pseudo-random noise (PN) sequences to be transmitted. The PN sequences for successive data symbols are concatenated, and the aggregate chip sequence is modulated onto the carrier using offset quadrature phase-shift-keying (O-QPSK). The functional block diagram in Figure 2.5 is provided as a reference for specifying the 2450 MHz PHY modulation and spreading functions. The number in each block refers to the subclause that describes that function. All binary data contained in the PPDU shall be encoded using the modulation and spreading functions shown in Figure 2.5. The 4 LSBs (b_0, b_1, b_2, b_3) of each octet shall map into one data symbol, and the 4 MSBs (b_4, b_5, b_6, b_7) of each octet shall map into next data

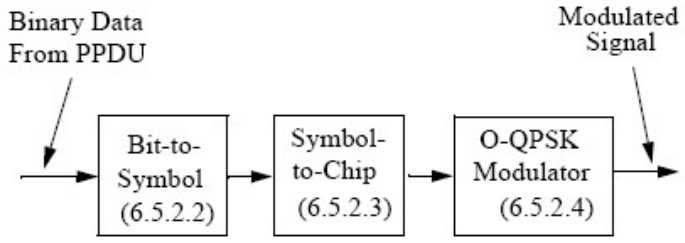

Figure 2.5: *Modulation and spreading functions[8]*

symbol. Each octet of the PPDU is processed through the modulation and spreading functions sequentially, beginning with Preamble field and ending with the last octet of the PSDU.

Each data symbol shall be mapped into a 32-chip PN sequence. The PN sequences are related to each other through cyclic shifts and/or conjugation (i.e., inversion of odd-indexed chip values).
The chip sequences representing each data symbol are modulated onto the carrier using O-QPSK with half-sine pulse shaping. Even-indexed chips are modulated onto the in-phase (I) carrier and odd-indexed chips are modulated onto the quadrature-phase (Q) carrier. Because each data symbol is represented by a 32-chip sequence, the chip rate (nominally 2.0 Mchip/s) is 32 times the symbol rate. To form the offset between I-phase and Q-phase chip modulation, the Q-phase chips shall be delated by T_c with the respect to the I-phase chips, where T_c is the inverse of the chip rate.

2.2.2 MAC sublayer

The MAC sublayer handles all access to the physical radio channel and is responsible for the following tasks:

- Generating network beacons if the device is a coordinator;
- Synchronization to network beacons;
- Supporting PAN association and disassociation;
- Supporting device security;

- Employing the CSMA-CA mechanism for channel access;
- Handling and maintaining the Guaranteed Time Slots (GTS) mechanism;
- Providing a reliable link between two peer MAC entities.

The MAC sublayer provides an interface between the SSCS and the PHY and conceptually includes a management entity called the MLME. This entity provides the service interfaces through which layer management functions may be invoked. The MLME is also responsible for maintaining a database of managed objects pertaining to the MAC sublayer. This database is referred to as the MAC sublayer PIB. Figure 2.6 depicts the components and interfaces of the MAC sublayer.

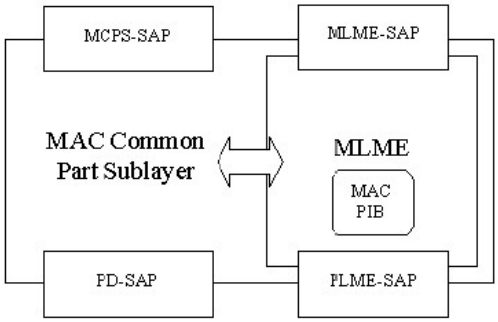

Figure 2.6: *The MAC sublayer reference model[8]*

The MAC sublayer provides two services: the MAC data service and the MAC management service interfacing to the MAC sublayer management (MLME) service access point (SAP) (known as MLME-SAP). The MAC data service enables the transmission and reception of MAC protocol data units (MPDUs) across the PHY data service.

General MAC frame format

The MAC frame format is composed of a MHR, a MAC payload , and a MFR. The fields of the MHR appear in a fixed order; however, the addressing fields may not be included in all frames. The general MAC frame shall be formatted as illustrated in Figure 2.7.

Octets: 2	1	0/2	0/2/8	0/2	0/2/8	0/5/6/10/14	variable	2
Frame Control	Sequence Number	Destination PAN Identifier	Destination Address	Source PAN Identifier	Source Address	Auxiliary Security Header	Frame Payload	FCS
		Addressing fields						
MHR							MAC Payload	MFR

Figure 2.7: *General MAC frame format*[8]

Frame Control field. The Frame Control Field is 2 octets in length and contains information defining the frame type, addressing fields, and other control flags, (Figure 2.8).

Bits: 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Addressing Mode	Frame Version	Source Addressing Mode

Figure 2.8: *Format of the Frame Control field*[8]

- *Frame Type subfield.* The Frame Type subfield is 3 bits in length and shall be set to one of the nonreserved values.
- *Security Enabled subfield.* The Security Enabled subfield is 1 bit in length, and it shall be set to one if the frame is protected by the MAC sublayer and shall be set to zero otherwise. The Auxiliary Security Header field of the MHR shall be present only if the Security Enabled subfield is set to one.
- *Frame Pending subfield.* The Frame Pending subfield is 1 bit in length and shall be set to one if the device sending the frame has more data for the recipient. This subfield shall be set to zero otherwise. The Frame Pending subfield shall be used only in beacon frames or frames transmitted either during the Contention Access Period (CAP) by devices

operating on a beacon-enabled PAN or at any time by devices operating on a nonbeacon-enabled PAN. At all other times, it shall be set to zero on transmission and ignored on reception.

- *Acknowledgment Request subfield.* The Acknowledgment Request subfield is 1 bit in length and specifies whether an acknowledgment is required from the recipient device on receipt of a data or MAC command frame. If this subfield is set to one, the recipient device shall send an acknowledgment frame only if, upon reception, the frame passes the third level of filtering. If this subfield is set to zero, the recipient device shall not send an acknowledgment frame.
- *PAN ID Compression subfield.* The PAN ID Compression subfield is 1 bit in length and specifies whether the MAC frame is to be sent containing only the one of the PAN identifier fields when both source and destination addresses are present. If this subfield is set to one and both the source and destination addresses are present, the frame shall contain only the Destination PAN Identifier field, and the Source PAN Identifier field shall be assumed equal to that of the destination. If this subfield is set to zero and both the source and destination addresses are present, the frame shall contain both the Source PAN Identifier and Destination PAN Identifier fields. If only one of the addresses is present, this subfield shall be set to zero, and the frame shall contain the PAN identifier field corresponding to the address. If neither address is present, this subfield shall be set to zero, and the frame shall not contain either PAN identifier field.
- *Destination Addressing Mode subfield.* The Destination Addressing Mode subfields is 2 bits in length and shall be set to one of the nonreserved values listed in Figure 2.9. If this subfield is equal to zero and the Frame Type subfield does not specify that this frame is acknowledgment or beacon frame, the Source Addressing Mode subfield shall be notzero, implying that the frame is directed to the PAN co-

Addressing mode value b_1 b_0	Description
00	PAN identifier and address fields are not present.
01	Reserved.
10	Address field contains a 16-bit short address.
11	Address field contains a 64-bit extended address.

Figure 2.9: *Possible value of the Destination Addressing Mode and Source Addressing Mode subfields*[8]

ordinator with the PAN identifier as specified in the Source PAN Identifier field.

- *Frame version subfield.* The Frame Version subfield is 2 bits in length and specifies the version number corresponding to the frame. This subfield shall be set to 0x00 to indicate a frame compatible with IEEE Std 802.15.4-2003 and 0x01 to indicate an IEEE 802.15.4 frame. All other values shall be reserved for future use.
- *Source Addressing Mode subfield.* The Source Addressing Mode subfield is 2 bits in length and shall be set to one of the nonreserved values listed in Figure 2.9.
  If the subfield is equal to zero and the Frame Type subfield does not specify that this frame is an acknowledgment frame, the Destination Addressing Mode subfiled shall be nonzero, implying that the frame has originated from the PAN coordinator with the PAN identifier as specified in the Destination PAN Identifier field.

Sequence Number field. The Sequence Number field is 1 octet in length and specifies the sequence identifier for the frame.
For a beacon frame, the Sequence Number field shall specify a BSN. For a data, acknowledgment, or MAC command frame, the Sequence Number field shall specify a DSN that is used to match an acknowledgment frame to the data or MAC command frame.

Destination PAN identifier field. The Destination PAN Identifier field, when present, is 2 octets in length and specifies the unique

PAN identifier of the intended recipient of the frame. A value of 0xFFFF in this field shall represent the broadcast PAN identifier, which shall be accepted as a valid PAN identifier by all devices currently listening to the channel.
This field shall be included in the MAC frame only if the Destination Addressing Mode subfield of the Frame Control field is nonzero.

Destination Address field. The Destination Address field, when present, is either 2 octets or 8 octets in length, according to the value specified in the Destination Addressing Mode subfield of the Frame Control field (Figure 2.8) and specifies the address of the intended recipient of the frame. A 16-bit value of 0xFFFF in this field shall represent the broadcast short address, which shall be accepted as a valid 16-bit short address by all devices currently listening to the channel.
This field shall be included to the MAC frame only if the Destination Addressing Mode subfiled of the Frame Control field is nonzero.

Source PAN Identifier field. The Source PAN Identifier filed, when present, is 2 octets in length and specifies the unique PAN identifier of the originator of the frame. This field shall be included in the MAC frame only if the Source Addressing Mode and PAN ID Compression subfields of the frame Control field are nonzero and equal to zero respectively.
The PAN identifier of a device is initially determined during association on a PAN, but may change following a PAN identifier conflict resolution.

Source Address field. The Source Address field, when present, is either 2 octets in length, according to the value specified in the Source Addressing Mode subfield of the Frame Control field (Figure 2.8), and specifies the address of the originator of the frame. This field shall be included in the MAC frame only if the Source Addressing Mode subfiled of the Frame Control field is nonzero.

Auxiliary Security header field. The Auxiliary Security Header

field has a variable length and specifies information required for security processing, including how the frame is actually protected (security level) and which keying material from the MAC security PIB is used. This field shall be present only if the Security Enabled subfield is set to one.

Frame Payload field. The Frame Payload field has a variable length and contains information specific to individual frame types. If the Security Enabled subfield is set to one in the Frame Control field, the frame payload is protected as defined by the security suite selected for that frame.

FCS field. The FCS field is 2 octets in length and contains a 16-bit ITU-T CRC. The FCS is calculated over the MHR and MAC payload parts of the frame.

2.3 Data transfer model

Three types of data transfer transactions exist. The first one is the data transfer to a coordinator in which a device transmits the data. The second transaction is the data transfer from a coordinator in which the device receives the data. The third transaction is the data transfer between two peer devices. In star topology, only two of these transactions are used because data may be exchanged only between the coordinator and a device. In a peer-to-peer topology, data may be exchanged between any two devices on the network; consequently all three transactions may be used in this topology. The mechanisms for each transfer type depend on whether the network supports the transmission of beacons. A beacon-enabled PAN is used in networks that either require synchronization or support for low-latency devices, such as PC peripherals. If the network does not need synchronization or support for low-latency devices, it can elect not to use the beacon for normal transfers. However, the beacon is still required for network discovery.

Data transfer to a coordinator

When a device wishes to transfer data to coordinator in a beacon-enabled PAN, it first listens for the network beacon. When the beacon is found, the device synchronizes to the superframe structure. At the appropriate time, the device transmits its data frame, using slotted CSMA-CA, to the coordinator. The coordinator may acknowledge the successful reception of the data by transmitting an optional acknowledgment frame. The sequence is summarized in Figure 2.10. When a

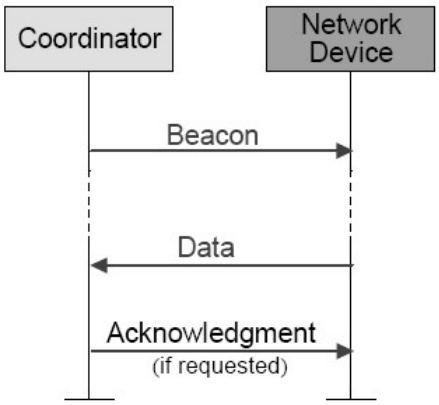

Figure 2.10: *Communication to a coordinator in a beacon-enabled PAN[8]*

device wishes to transfer data in a nonbeacon-enabled PAN, it simply transmits its data frame, using unslotted CSMA-CA, to the coordinator. The coordinator acknowledges the successful reception of the data by transmitting an optional acknowledgment frame. The transaction is now completed. The sequence is summarized in Figure 2.11.

Data transfer from a coordinator

When the coordinator wishes to transfer data to a device in a beacon-enabled PAN, it indicates in the network beacon that the data message is pending. The device periodically listens to the network beacon and, if a message is pending, transmits a MAC command requesting the data, using slotted CSMA-CA. The coordinator acknowledges the

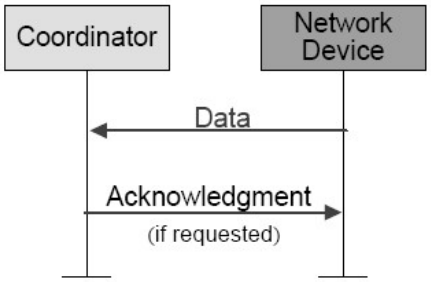

Figure 2.11: *Communication to a coordinator in a nonbeacon enabled PAN [8]*

successful reception of the data request by transmitting an acknowledgment frame. The pending data frame is then sent using slotted CSMA-CA or, if possible, immediately after the acknowledgment. The device may acknowledge the successful reception of the data by transmitting an optional acknowledgment frame. The transaction is now complete. Upon successful completion of the data transaction, the message is removed from the list of pending messages in the beacon. This sequence is summarized in Figure 2.12. When a coordinator

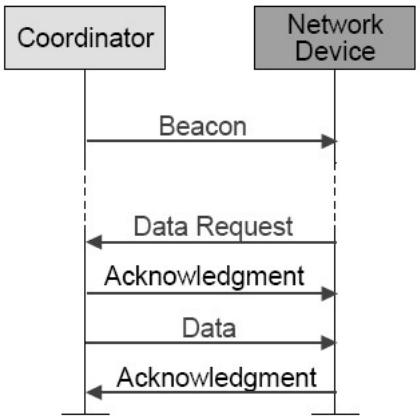

Figure 2.12: *Communication from a coordinator in a beacon-enabled PAN[8]*

wishes to transfer data to device in a nonbeacon-enabled PAN, it stores

the data for the appropriate device to make contact and request the data. A device may make contract by transmitting a MAC command requesting the data, using unslotted CSMA-CA, to its coordinator at an application defined rate. The coordinator acknowledges the successful reception od the data request by transmitting an acknowledgment frame. If a data frame is pending, the coordinator transmits the data frame, using unslotted CSMA-CA, to the device. If a data frame is not pending, the coordinator indicates this fact either in the acknowledgment frame following the data request or in a data frame with zero-length payload. If requested, the device acknowledges the successful reception of the data frame by transmitting an acknowledgment frame. This sequence is summarized in Figure 2.13.

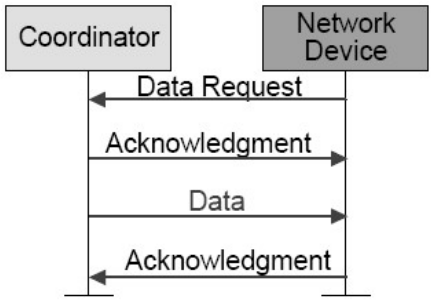

Figure 2.13: *Communication form a coordinator in a nonbeacon-enabled PAN[8]*

2.3.1 Frame structure

The frame structure have been designed to keep the complexity to a minimum while at the same time making them sufficiently robust for transmission on a noisy channel. Each successive protocol layer adds to the structure layer-specific headers and footers. This standard defines four frame structures:

- A **Beacon frame**, used by a coordinator to transmit beacons;
- A **Data frame**, used for all transfers of data;

- An **Acknowledgment frame**, used for confirming successful frame reception;
- A **MAC command frame**, used for handling all MAC peer entity transfers.

2.3.2 Beacon frame

Figure 2.14 shows the structure of the beacon frame, which originates from within the MAC sublayer. A coordinator can transmit network beacons in a beacon-enabled PAN. The MAC payload contains the superframe specifications, GTS fields, pending address fields, and beacon payload. The MAC payload is prefixed with a MAC header (MHR)(previously mentioned) and appended with a MAC footer (MFR). The MHR contains the MAC Frame Control field, beacon sequence number (BSN), addressing fields, and optionally the auxiliary security header. The MFR contains a 16-bit frame check sequence (FCS). The MHR, MAC payload, and MFR together form the MAC beacon frame (i.e., MPDU). The MAC beacon frame is then passed to the PHY as the

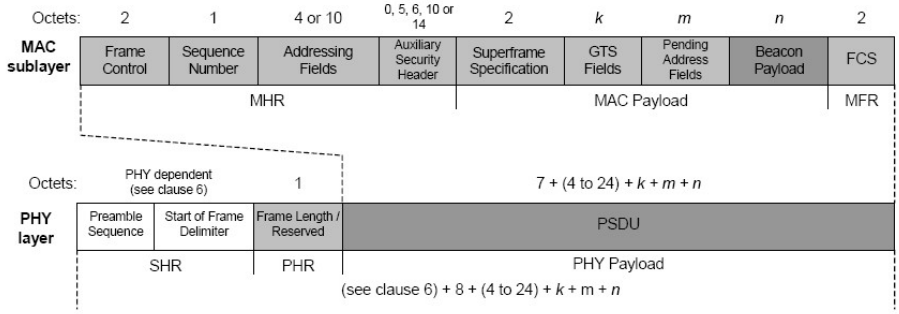

Figure 2.14: *Schematic view of the beacon frame and the PHY packet[8]*

PHY service data unit (PSDU), which becomes the PHY payload. The PHY payload is prefixed with a synchronization header (SHR), containing the Preamble Sequence and Start-of-Frame Delimiter (SFD) fields, and a PHY header (PHR) containing the length of the PHY payload in octets. The SHR, PHR, and PHY payload together form the PHY packet(i.e., PPDU).

2.3.3 Data frame

Figure 2.15 shows the structure of the data frame, which originates from the upper layers. The data payload is passed to the MAC sub-

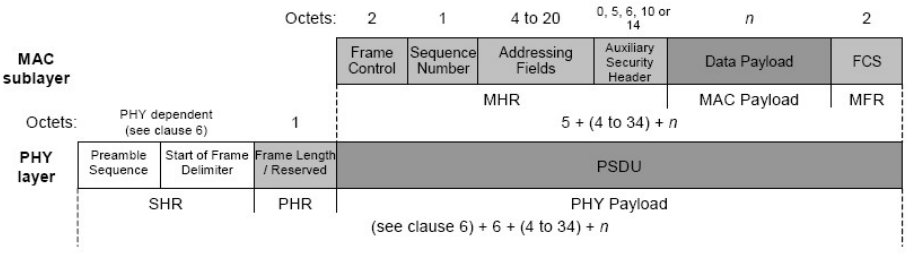

Figure 2.15: *Schematic view of the data frame and the PHY packet[8]*

layer and is referred to as MAC service data unit (MSDU). The MAC payload is prefixed with an MHR and appended with an MFR. The MHR contains the Frame Control field, data sequence number (DSN), addressing fields, and optionally the auxiliary security header. The MFR is composed of a 16-bit FCS. The MHR, MAC payload, and MFR together form the MAC data frame (i.e., MPDU).

The MPDU is passed to the PHY as the PSDU, which becomes the PHY payload. The PHY payload is prefixed with an SHR, containing the Preamble Sequence and SFD fields, and a PHR containing the length of the PHY payload in octets. The SHR, PHR, and PHY payload together form the PHY packet, (i.e., PPDU).

2.3.4 Acknowledgment frame

Figure 2.16 shows the structure of the acknowledgment frame Acknowledgment Frame (ACK), which originates from within the MAC sublayer. The MAC acknowledgment frame is constructed from an MHR and an MFR; it has no MAC payload. The MHR contains the MAC Frame Control field and DSN. The MFR is composed of a 16-bit FCS. The MHR and MFR together form the MAC acknowledgment frame (i.e.; MPDU).
The MPDU is passed to the PHY as the PSDU, which becomes the

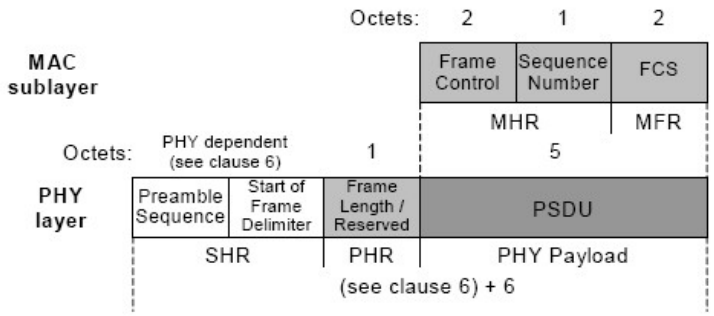

Figure 2.16: *Schematic view of the acknowledgment frame and the PHY packet[8]*

PHY payload. The PHY payload is prefixed with the SHR, containing the Preamble Sequence and SFD fields, and the PHR containing the length of the PHY payload in octets. The SHR, PHR, and PHY payload together form the PHY packet, (i.e., PPDU).

2.3.5 MAC command frame

Figure 2.17 shows the structure of the MAC command frame, which originates from within the MAC sublayer. The MAC payload contains the Command Type field and the command payload. The MAC payload is prefixed with an MHR and appended with an MFR. The MHR contains the MAC Frame Control field, DSN, addressing fields, and optionally the auxiliary security header. The MFR contains a 16-bit FCS. The MHR, MAC payload, and MFR together form the MAC command frame (i.e.; MPDU). The MPDU is than passed to the PHY as the PSDU, which becomes the PHY payload. The PHY payload is prefixed with an SHR, containing the Preamble Sequence and SFD fields, and a PHR containing the length of the PHY payload in octets. The preamble sequence enables the receiver to achieve symbol synchronization. The SHR, PHR, and PHY payload together form the PHY packet, (i.e.; PPDU).

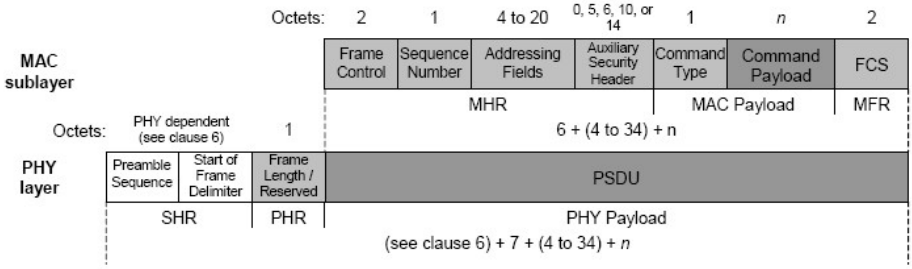

Figure 2.17: *Schematic view of the MAC command frame and the PHY packet[8]*

2.4 ZigBee higher levels overview

ZigBee technology

Zigbee wireless technology is a short-range communication system for applications with relaxed throughput and latency requirements in wireless personal area networks. The key features of Zigbee wireless technology are low complexity, low cost, low power consumption, low data rate transmissions, supported by cheap fixed or moving devices. The main field of application of this technology is the implementation of WSNs.
The IEEE 802.15.4 Working Group [2] focuses on the standardization of the bottom two layers of the ISO/OSI protocol stack. The other layers are normally specified by industrial consortia such as ZigBee Alliance [3]. The purpose of the Zigbee Alliance is to univocally describe the ZigBee protocol standard in such a way that interoperability is guaranteed also among devices produced by different companies, provided that each device implements the ZigBee protocol stack. The ZigBee architecture is composed of set of blocks called layers, as depicted in Figure 2.18. Each layer performs a specific set of services for the layer above.

Given the IEEE 802.15.4 specifications on PHY and MAC layer, the ZigBee Alliance defines the network layer and the framework for the application layer. The responsibilities of ZigBee network layer in-

[2] http://www.ieee802.org
[3] ZigBee Alliance: http://www.zigbee.org

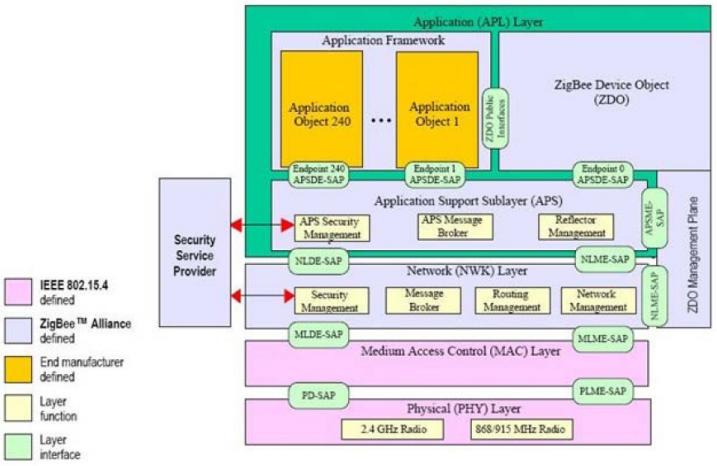

Figure 2.18: *Architecture of the ZigBee stack[16]*

clude: mechanisms to join and leave the network, frame security, routing, path discovery, one-hop neighbors discovery, neighbor information storage. The ZigBee application layer consists of the application support sublayer, the application framework, the Zigbee device object, and the manufacturer-defined application objects. The responsibilities of the application support sublayer include: maintaining tables for binding (defined as the ability to match two devices together based on their services and their needs) and forwarding messages between bound devices. The responsibilities of the Zigbee device objects include: defining the role of the device within the network (e.g., PAN coordinator or end device), initiating and/or responding to binding requests, establishing secure relationships between network devices, discovering devices in the network, and determining which application services they provide.

Chapter 3

Packet Correcting Schemes in Wireless Sensor Networks

This Chapter regards the introduction of the Packet Correcting Schemes in the WSNs in order to save the energy spend for the retransmission of the packets erased by the spent transmission channel . In the first section are defined the three fundamental recovery schemes: *Automatic Repeat ReQuest* (ARQ), *Forward Error Correction* (FEC) and Hybrid Automatic Repeat and Request (HARQ) used in WSNs. In the following sections are studied and discussed the block diagram from the transmission chain, fundamental concepts and definitions of the coding theory, codes implemented in the developed applications and the encoding/decoding techniques that make use of these codes. In the last section is described the implementation of a software simulator of the entire transmission scenario in order to confirm the theoretical expectations. Finally, the simulation results are presented.

3.1 Packet Erasure Correcting Schemes

As opposed to wired channels, wireless channels often have a poorer quality in terms of bit/symbol error rate. The actual channel quality depends on many factors, including frequency, distance between transmitter and receiver, and their relative speed, propagation environment (number of paths and their respective attenuation), technology, and much more. Physical phenomena like reflection, diffraction, and scat-

tering of waveforms, partially in conjunction with moving nodes or movements in the environment, lead to fast fading and intersymbol interference. Path loss, attenuation, and the presence of obstacles lead to slow fading. In addition, there is noise and interference from other nodes/other systems working in overlapping or neighboring frequency bands. The distortion of waveforms translates into bit errors and packet losses.
The bit-error and packet-loss statistics also depend on modulation scheme and presence of interferers. Several studies about these statistics show some common properties:

- Both bit errors and packet losses are "bursty", that is they tend to occur in clusters with error-free periods ("runs") between the clusters. The empirical distributions of the cluster and run lengths often have a large coefficient of variation or sometimes even seem to be heavy tailed.

- The error behavior even for stationary transmitters and receivers is time varying, and the instantaneous bit-error rates can be sometimes quite high. The same is true for packet-loss rates.

The bursty nature of wireless channel errors is a source of both problems. To improve the transmission quality on wireless channels is possible by working on the physical as well as on higher layers.

It is important to use some error correction schemes in order to control the errors introduced and to reduce the number of automatic requests for retransmission. As it was mentioned in Chapter 1, the sensor nodes are typically wireless nodes with limited storage and computational power. Thus the error control schemes should be energy efficient at sensor nodes level. Due to the rigorous energy consumption constraints, minimization of transmission power is extremely important in WSNs. Reduction of the transmission power decreases the Packet Delivery Ratio (PDR) [2] due to the nature of the radio environment such that fewer packets can be received. However, lower signal to noise ratio can be compensated by Packet Correcting Codes (PCC) and thus, reliability of packet transmissions can be improved. On the other hand, efficient packet coding allows longer hop distances with the same transmission power while sufficient PDR is maintained. For every application, a maximum Bit Error Rate (BER) or Packet Delivery

Ratio (PDR) is specified to achieve a certain Quality of Service (QoS). To maintain the BER/PDR within this limit, either the transmit signal power can be increased or packet correcting codes (PCC) can be used. PCC reduces the required transmitted signal energy because of the coding gain. Incorporating PCC results in additional energy consumption due to distinct two factors: transmitting "redundant" packets and computation energy required for encoding/decoding. For energy optimal designs, PCC can be used when the energy saving due to the coding gain more than compensates the additional energy spent in transmitting the redundant bits as well as the energy spent in the process of encoding/decoding. Intuitively, for very short distance transmission, using PCC may not be energy efficient as the energy overheads are likely to be more than the energy savings. As the distance increases, PCC will become energy efficient as the coding gain will keep the transmitted power low for the same BER/PDR.

In wireless communication systems, packet correction schemes can be divided into three categories based on operation principles:

1. *Automatic Repeat ReQuest* (ARQ);
2. *Forward Error Correction* (FEC);
3. *Hybrid Automatic Repeat ReQuest* (HARQ);

If a packet transmission fails for some reason and the packet cannot be decoded properly at the receiver, the straightforward solution is to retransmit the entire packet again. This kind of approach is called ARQ. It is very simple to use but the disadvantage of using it is the additional retransmission energy cost and area overhead [26]. HARQ that combines ARQ and FEC is even worse [27] since it consumes a lot of energy and is limited to some specific applications. Hybrid ARQ schemes aim to improve reliability by adding redundant bits or packets in an incremental fashion depending on the number of experienced packet losses.
The purpose of FEC approach is to enhance error resiliency by introducing redundant information such that decoding is possible even through some bits or packets are misinterpreted or lost. The main advantage with FEC is that there are no delays in message flows, through the packet might get lost if the packet correction scheme is not strong

enough. The limited battery capacity of each sensor node makes the minimization of the power consumption one of the primary concern in WSN in order to increase the entire network lifetime.
Energy constrained transmission issue of WSN makes FEC a popular technique to be used in such networks rather than ARQ and HARQ. For most of the codes used in WSN, encoding is simple and energy consumption is low [28]. However decoding part is usually complex and it consumes a significant amount of energy. Decoding being done at every node, results in to higher energy consumption at every node which reduces the network life in turn. When applying any of these schemes in data transmission via WSN, an important question needs to be answered: What is the probability of reproducing the original data overcoming all the erasures introduces by noisy channel while transmision? At Chapter 5, thanks to the experimental results, it will be possible to answer that question.

3.2 Characterization of the transmission chain

The history of PCC started with the introduction of the Hamming codes [Ham], at about the same time as the seminal work of Shannon [Sha]. Figure 3.1 shows the block diagram of a canonical digital communications/storage system. In the following sections are described each component of the transmission chain and its functionality in the entire communication system.

3.2.1 Characterization of the encoder/decoder

In Figure 3.1 the information source and destination include any source coding scheme matched to the information. The PCC encoder takes as input the information packets from the source and adds redundant packets to it, so that most of the erasure packets - introduced in the process of modulating a signal or transmitting it over a noisy medium, can be recovered. At the receiver end, the PCC decoder utilizes the redundant packets to correct possible channel packet erasures. In the case of packet erasure detection, the decoder can be thought of as a

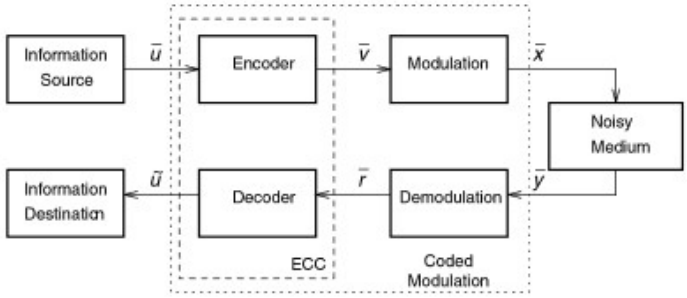

Figure 3.1: *A canonical digital communications system[20]*

re-encoder of the received message and a check that the redundant packets generated are the same as those received.

3.2.2 Packet correcting coding: Basic concepts

All packet correcting codes are based on the same basic principle: Redundancy is added to information in order to correct any errors or erasures that may occur in the process of storage or transmission. In a basic (and practical) form, redundant symbols are appended to information symbols to obtain a coded sequence or codeword. A codeword obtained by encoding with a *block code* is shown in Figure 3.2.

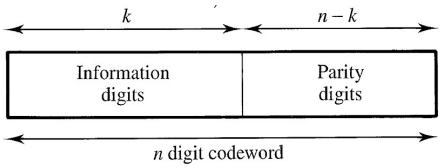

Figure 3.2: *Systematic block encoding for error correction[20]*

Such an encoding is said to be systematic. This means that the information symbols always appear in the first k positions of a codeword. The remaining $n-k$ symbols in a codeword are some function of the information symbols, and provide redundancy that can be used for error correction/detection purposes. The set of all code sequences is called an *error correcting code*.
WSN can benefit from special coding schemes: erasure correcting codes.

Using erasure codes, it is possible the reconstruction of the k original messages with the usage also of the $n-k$ redundancy messages. Erasure code relives constraints on packet loss distribution. Since erasure code does not require reverse path, it easily increases reliability of convergence routing, where there is no backward path. Figure 3.3 shows high level mechanisms of erasure code. In this thesis, two different

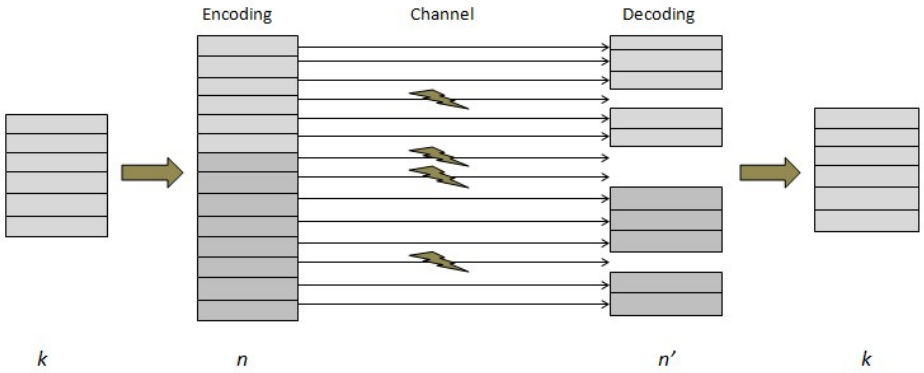

Figure 3.3: *Mechanism of erasure code[25]*

types of codes are considerate for implementation in WSN nodes:

- Single Check Parity Code;

- Hamming Code;

Both types of these codes are *linear codes*.

Basics

If C is a linear code that, as a vector space over the field F, has dimension k, than the C is an $[n,k]$ linear code over F. In particular, the rate of $[n,k]$ linear code is k/n. If C has minimum distance d, then C is an $[n,k,d]$ linear code over F. The number of $n-k$ is again the *redundancy* of C.
Consider an error correcting code C with binary elements. In order to achieve error correcting capabilities, not all the 2^n possible binary vectors of length n are allowed to be transmitted. Instead, C is a

subset of the n-dimensional binary vector space $V_2 = \{0,1\}^n$, such that its elements are as far as possible. In the binary space V_2, *distance* is defined as the number of entries in which two vectors differ.
Let $\overline{x}_1 = (x_{1,0}, x_{1,1}, ..., x_{1,n-1})$ and $\overline{x}_2 = (x_{2,0}, x_{2,1}, ..., x_{2,n-1})$ be two vectors in V_2. Than the ***Hamming distance*** between $\overline{x}_1$ and $\overline{x}_2$, denoted $d_H(\overline{x}_1, \overline{x}_2)$ is defined as:

$$d_H(\overline{x}_1, \overline{x}_2) = \mid \{i : x_{1,i} \neq x_{2,i}\} \mid,$$

where $\mid A \mid$ denotes the number of elements in (or the cardinality of) a set A.
Given a code C, its ***minimum Hamming distance***, d_{min}, is defined as the minimum Hamming distance among all possible distinct pairs of codewords in C.

$$d_{min} = \min_{\overline{v}_1, \overline{v}_2 \in C} \{d_H(\overline{v}_1, \overline{v}_2) \mid \overline{v}_1 \neq \overline{v}_2\}$$

The binary vector space V_2 is also known as a *Hamming space*. Let $\overline{v}$ denote a codeword of an error correcting code C. A *Hamming sphere* $S_t(\overline{v})$, of radius t and centered around $\overline{v}$, is the set of vectors in V_2 at a distance less than or equal to t from the center $\overline{v}$,

$$S_t(\overline{v}) = \{\overline{x} \in V_2 \mid d_H(\overline{x}, \overline{v}) \leq t\}.$$

The size of (or the number of codewords) in $S_t(\overline{v})$ is given by the following expression

$$\mid S_t(\overline{v}) \mid = \sum_{i=0}^{t} \binom{n}{i}.$$

The ***error correcting capability***, t, of a code C is the largest radius of Hamming spheres $S_t(\overline{v})$ around all the codewords $\overline{v} \in C$, such that for all different pairs $\overline{v}_i, \overline{v}_j \in C$, the corresponding Hamming spheres are disjoint, i.e.,

$$t = \max_{\overline{v}_i, \overline{v}_j \in C} \{l \mid S_l(\overline{v}_i) \cap S_l(\overline{v}_j) = \oslash, \overline{v}_i \neq \overline{v}_j\}$$

In terms of the minimum distance of C, d_{min}, an equivalent and more common definition is

$$t = \lfloor (d_{min} - 1)/2 \rfloor,$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x.
One of the advantages of *linear* block codes is that the computation of d_{min} requires one only to know the *Hamming weight* of all $2^k - 1$ nonzero codewords. Linear codes are vector subspaces of V_2. This means that the encoding can be accomplished by matrix multiplications. In terms of digital circuitry, simple encoders can be built using exclusive OR's, AND gates and D flip-flops.

Generator and Parity-check matrices

Let C denote a binary linear (n, k, d_{min}) code. Since C is a k-dimensional vector subspace, it has a *basis*, say $\{\overline{v}_0, \overline{v}_1, ..., \overline{v}_{k-1}\}$, such that any codeword $\overline{v} \in C$ can be represented as a linear combination of the elements in the basis:

$$\overline{v} = u_0\overline{v}_0 + u_1\overline{v}_1 + ... + u_{k-1}\overline{v}_{k-1}$$

where $u_i \in \{0, 1\}, 1 \leq i < k$. This equation could be written in terms of a *generator matrix* G and a message vector, $\overline{u} = (u_0, u_1, ..., u_{k-1})$ as follows:

$$\overline{v} = \overline{u}G,$$

where

$$G = \begin{pmatrix} \overline{v}_0 \\ \overline{v}_1 \\ . \\ . \\ . \\ \overline{v}_{k-1} \end{pmatrix} = \begin{pmatrix} v_{0,0} & v_{0,1} & ... & v_{0,n-1} \\ v_{1,0} & v_{1,1} & ... & v_{1,n-1} \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ v_{k-1,0} & v_{k-1,1} & ... & v_{k-1,n-1} \end{pmatrix}$$

Since C is a k-dimensional vector space in V_2, there is an $n - k$-dimensional *dual space* $C^\top$, generated by the rows of a matrix H, called *parity-check matrix*, such that $GH^\top = 0$, where $H^\top$ denotes the transpose of H. In particular, for any codeword $\overline{v} \in C$,

$$\overline{v}H^\top = \overline{0}.$$

As it was mentioned, a nice feature of linear codes is that computing the minimum distance of the code amounts to computing the minimum Hamming weight of its nonzero codewords. In this section, this fact is

shown. First, define the *Hamming weight*, $w_H(\overline{x})$, of a vector $\overline{x} \in V_2$ as the number of nonzero elements in $\overline{x}$. From the definition of the Hamming distance, it is easy to see that $w_H(\overline{x}) = d_(\overline{x}, \overline{0})$. For a binary liner code C, note that the distance

$$d_H(\overline{v}_1, \overline{v}_2) = d_H(\overline{v}_1 + \overline{v}_2, \overline{0}) = w_H(\overline{v}_1 + \overline{v}_2).$$

Finally, by linearity, $\overline{v}_1 + \overline{v}_2 \in C$. As a consequence, the minimum distance of C can be computed by finding the minimum Hamming weight among the $2^k - 1$ nonzero codewords. Let see in detail the main characteristics of the codes used and the relative encoding/decoding process.

3.2.3 Single Check Parity Code

A single-check parity code is one of the most common forms of detecting transmission errors. This code uses one extra packet in a block of k packets to indicate whether the number of 1s in a block is odd or even. Thus, if a single error occurs, either the parity packet is corrupted or the number of detected 1s in the information bit sequence will be different from the number used to compute the parity packet. In either case the parity packet will not correspond to the number of detected 1s in the information bit sequence so the single error is detected. When used on Binary Erasure Channel Binary Erasure Channel (BEC), it gives the possibility to correct one packet erasure.
One generator matrix for an code $(n, n-1)$ (SPC code (e.g. for $n = 4$)) is the following:

$$G_{parity} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Moreover, its parity matrix is:

$$H_{parity} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix}$$

It is possible to find the minimum distance of a linear code from the parity-check matrix H. The minimum distance is equal to the smallest number of linearly-dependent columns of H. A vector v is a codeword if $\boldsymbol{v}H^\top = \overline{0}$. If d columns of H are linearly dependent, let $\boldsymbol{v}$ have 1's

in those positions, and 0's elsewhere. Thus $\boldsymbol{v}$ is a codeword of weight d. And if there were any codeword of weight less than d, the 1s in that codeword would identify a set of less than d linearly-dependent columns of H. For example, if H has a column of all zeros, than d=1, if H has two identical columns, than $d \leq 2$. For binary codes, if all columns are distinct and non-zero, than $d \geq 3$.
In this case, it is evident that the minimum distance of this code is $d = 2$, hence the number of detecting errors is $t = 1$, which is also the number of correctable erasures. Applying $\overline{u}G = \overline{v}$, it is possible to obtain the codewords:

$$\begin{pmatrix} u_0 & u_1 & u_2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_0 + u_1 + u_2 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

The first three codewords are exactly the information messages, and the fourth is the parity codeword. The parity-check matrix H and the relationship $\overline{v}H^{\top} = 0$ allow the formation of the system equation that has to be satisfied at the receiver side:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = 0$$

Applying the encoding process it is possible obtain the codewords. As it was shown, correction capabilities of this code allow the detection of one error in the transmission process. In case of one packet erasure, it is possible the recovery of that packet simply applying the xor-operation (sum mod 2) between the remaining packets received. Figure 3.4 illustrates the encoding and decoding process.

3.2.4 Hamming Code

Hamming Codes are a family of $[n, k]$ *linear block codes* that have the following parameters

$$\begin{aligned} n &= 2^m - 1 \\ k &= 2^m - m - 1 \end{aligned}$$

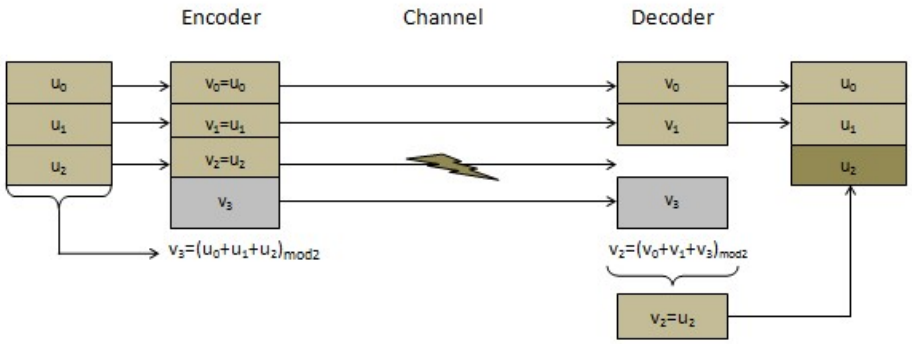

Figure 3.4: *Encoding/decoding using Single Check Parity Code*

Hamming codes have a minimum distance $d_{min} = 3$, and thus are single-error correcting codes and double erasure correcting. The generator and parity-check matrices in case of Hamming Code (7,4) are the following:

$$G_{hamming} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

$$H_{hamming} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

In case of packet transmissions, the encoding permits to obtain the redundancy packets simply from the $\overline{u}G_{hamming} = \overline{v}$.

$$\begin{pmatrix} u_0 & u_1 & u_2 & u_3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_0 + u_1 + u_2 \\ u_0 + u_1 + u_3 \\ u_0 + u_2 + u_3 \end{pmatrix} = \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{pmatrix}$$

The encoder adds three redundancy parity packets v_4, v_5, and v_6, which permit the recovery of possible erasure packets at the receiver.

The decoder after the reception of the packets, uses the parity-check matrix to determinate the correctness of the received packets. The relation $\overline{v}H^{\top} = 0$ determinants the resolution of a system of parity equation:

$$\begin{cases} u_0 + u_1 + u_2 + v_4 &= 0 \\ u_0 + u_1 + u_3 + v_5 &= 0 \\ u_0 + u_2 + u_3 + v_6 &= 0 \end{cases}$$

Every time when the transmission is affected by one or two packet erasures, the decoder can recover the missing packets simply by resolving this system of equation. Figure 3.5 shows the encoding and decoding algorithms using this type of code.

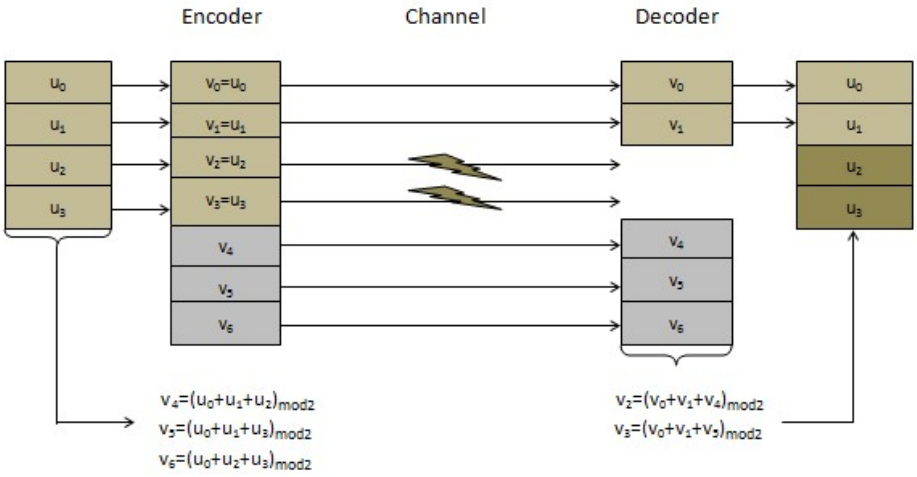

Figure 3.5: *Encoding and decoding using Hamming(7,4) Code*

3.2.5 Characterization of the transmission channel

Information theory provides core channel models that are used to represent a wide range of communication and networking scenarios. In this case a basic link through two sensor nodes could be illustrated with a Discrete Memoryless Channel (DMC) model. A DMC is characterized by the relationship between its input X and its output Y, where X and Y are two (hopefully) dependent random variables. Therefore, a DMC is usually represented by the conditional probability *p(y|x)* of the channel output Y given the channel input X. Furthermore, and since X and Y are dependent on each other, their mutual information *I(X;Y)* has a nonzero (i.e., strictly positive) value,

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \frac{p(x,y)}{p(x)\, p(y)} > 0.$$

An important measure is the maximum amount of information that Y can provide about X for a given *p(y|x)*. This measure can be evaluated by maximizing the mutual information *I(X;Y)* over all possible sources characterized by the marginal probability mass function *p(x)* of the channel input X. This maximum measure of the mutual information is known as the "information" channel capacity C:

$$C = \max_{p(x)} I(X;Y).$$

Based on this definition, the channel capacity C is a function of the parameters that characterize the conditional probability *p(y|x)* between the channel input X and the channel output Y. The following section describe a particular channel of interest.

The Binary Erasure Channel (BEC)

The simplest DMC channel model that could be used for representing a link or route in a WSN is the Binary Erasure Channel (Figure 3.6). The BEC is characterized by the following:

- The input X is a binary (Bernoulli) random variable that can be either a zero or a one.

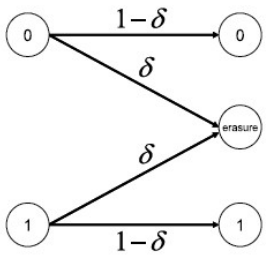

Figure 3.6: *A representation of the Binary Erasure Channel [29]*

- A loss parameter δ, which represents the probability that the input is lost ("erased" or "deleted") when transmitted over the BEC channel.
- The output Y is a ternary random variable that could take on one of three possible values: zero, one or "erasure". The latter output occurs when the channel loses the transmitted input X.

More specifically, a BEC is characterized by the conditional probability measures:

$$Pr[Y = \text{erasure} \mid X = 0] = \delta \,\text{and} Pr[Y = \text{erasure} \mid X = 1] = \delta$$

,

$$Pr[Y = 0 \mid X = 0] = 1 - \delta \,\text{and}\, Pr[Y = 1 \mid X = 1] = 1 - \delta$$

,

$$Pr[Y = 1 \mid X = 0] = 0 \,\text{and}\, Pr[Y = 0 \mid X = 1] = 0.$$

Therefore, no errors occur over a BEC, as $Pr[Y = 0 \mid X = 1] = Pr[Y = 1 \mid X = 0] = 0$. Due to the loss symmetry of the BEC (i.e., the conditional probability of losing a bit is independent of the bit value), it can be easily shown that the overall loss probability is also the parameter δ. In other words,

$$Pr[Y = \text{erasure}] = \delta.$$

By using the definition of information channel capacity, it can be shown that the channel capacity of the BEC is a rather intuitive expression,

$$C = 1 - \delta.$$

This capacity, which is measured in "bits" per "channel use", can be achieved when the channel input X is a uniform random variable with $Pr[X = 0] = Pr[X = 1] = 1/2$.

The "Packet" Erasure Channel (PEC)

A simple generalization of the BEC is needed to capture the fact that the information content to transmit is usually packetized and transmitted over wireless links as "integrated vectors" of bits rather than individual bits. In other words, when a data packet is lost, that packet is lost in its totality. Hence, for bits that belong to the same packet, these bits are dependent on each other: either all the bits are transmitted successfully (usually without errors) or all the bits are erased. In this sense was done this generalizations as Packet Erasure Channel (PEC). In this case, the input is a vector of random variables: $\underline{X}=(X_1, X_2,...,X_n)$, where each element X_i is a binary random variable. The output of the channel includes the possible "erasure" outcome and all possible input vectors. In other words, the following conditional probability measures for the PEC are valid:

$$Pr[\underline{Y} = \text{erasure} \mid \underline{X}] = \delta$$

$$Pr[\underline{Y} = \underline{X} \mid \underline{X}] = 1 - \delta$$

Note that these conditional probability measures are independent of the particular input vector $\underline{X}$ (i.e., packet). Consequently, it is not difficult to show that the PEC has the same basic measures, such as channel capacity, as the BEC. Therefore, $C = 1 - \delta$. The capacity in this case is measured in "packets" per "channel use".
Regards this thesis, it is better to refer to this type of channel, considering the fact that the implementation of the algorithms was made using packets transmissions.

In this context, it is useful to define some quality of service parameters because their analysis give insight to the mean behavior of the network. One of this figures of merit is the PER (Packet Erasure Ratio) [2]. This parameter represents the number of incorrectly received data packets divided by the total number of transmitted packets. The expectation value of the PER is denoted as *Packet Error Probability*.

For a transmission of n packets through the BEC (PEC) channel characterized by independent losses and erasure probability p, the probability of loss p and without introducing any encoding/decoding techniques in the transmission chain, it can be expressed as:

$$P_e = 1 - (1 - p)^n$$

In the real-time applications realized in this thesis, the PER parameter was chosen as a parameter to simulate and through which it was possible to verify the behavior of the wireless network created, confirm the theoretical expectations and verify the correctness of the various algorithms implemented.

3.3 The simulation of PER

Before the implementation of real-time applications, it was implemented a software simulator capable to model a transmission of packets through a memoryless PEC channel characterized by a probability loss (p), realized using the *Monte Carlo* method. The aim of the simulation is to validate the theoretical expectations and estimate the evolution of the probability of error P_e in case encoding/decodng technique is introduced in the transmission chain. In this case, it was considered the *Single Check Parity Code*. The simulation provides multiples of 10 packet transmissions, where the 10^{th} is a redundancy packet calculated as a parity packet of the previous 9 data packets. The parity code allows the recovery of any single lost data packet that belongs at a specific source block present at the sender node. Since the *Single Parity Check Code* give the opportunity for the recovery of only one packet erasure during the transmission process, more than two packet erasures cannot be recovered.
If p is the probability of error for a single packet over a PEC, than the probability of error is:

$$\begin{aligned} P[\text{uncorrectable error}] &= P[\text{2 or more packet erasures}] \\ &= 1 - P[\text{1 or less packet erasures}] \\ &= 1 - P[\text{no erasures}] - P[\text{1 erasure}] \\ &= 1 - (1 - p)^{10} - 10(1 - p)^9 p \end{aligned}$$

The theoretical evolution of this *Probability of Error* is illustrated on Figure 3.7. In the simulation of this transmission scenario, the *Prob-*

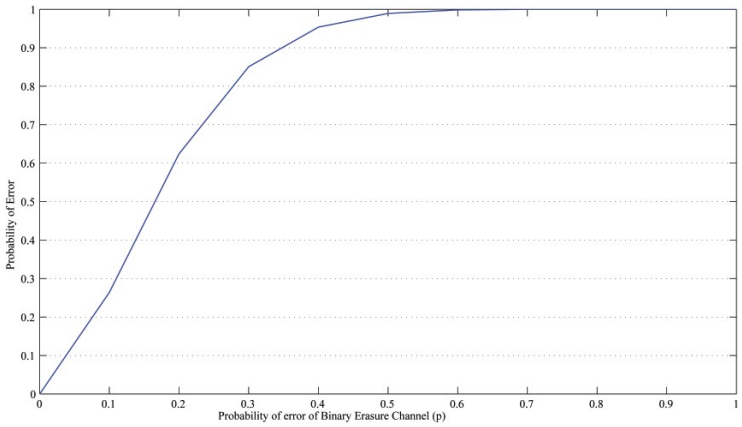

Figure 3.7: *Probability of error using Single Check Parity Code*

ability of error was evaluated, as mentioned before, through the PER which is calculated as ratio between the number of failed decodings and the total number of transmissions. The expectation value of the PER is the *Packet Error Probability*. It is verified that the approximation of the calculated PER value tends to the *Probability of Error* for quite high values of the number of failed decodings (for example $num\_decod = 100$) . Thus, the simulation parameter (PER) is calculated when the number of failed decoding exceed its limit value (100 failed decodings).

3.3.1 Simulation parameters and flow diagram

The simulation prevents transmission of 10 packets through a PEC channel whose probability loss value varies from p=0 to p=1. For the realization of the transmission model of the BEC channel, it was introduces a random variable n, which assumes uniformly distributed values in the interval [0,1] and is compared to the probability of the channel p. In this way, every time that the n variable results less than the p, the packet is considered as erased packet during the transmission, otherwise is considered as successfully received packet. Given

that the PER is calculated as the ratio of the number of failed decoding process and the total number of transmission of a 10 packet blocks, it is necessary the introduction of variables that are used to count the times when the transmission finished with success and others used to count the times when packet erasures were detected. The variables used for this scope are:

- *num_dec*. Variable used to count the number of times in which the decoding process failed.
- *count_error_10*. Variable used to count the number of erasures verified in 10 packets transmission;
- *count_error_glob*. Variable used to count the number of erasures verified until the num_dec variable has not exceed its limit;
- *count_correct_10*. Variable used to count the number of successfully arrived packets in 10 packets transmission.
- *count_correct_glob*. Variable used to count the number of successfully arrived packets until the num_dec variable has not exceed its limit;

Figure 3.8 shows the flow diagram of the simulation.

The simulation considers the transmission of multiplies of source blocks each composed of 10 packets. Then, for every packet the random variable n assumes value in the interval [0,1]. If the value result less than p, the packet is considered erased and the number of *count_error_10* is increased, otherwise , the *count_correct_10* update its value. The 10^{th} packet is calculated as redundancy parity packet and its introduction serves to prevent a possible packet erasure. The introduction of the parity code is capable to recuperate only one packet erasure at the receiver side. Thus, after the reception of entire source block, depending of the value of *count_error_10*, it is possible distinguish 3 different situations:

- *count_error_10* value is 0; transmission with all packets successfully received;

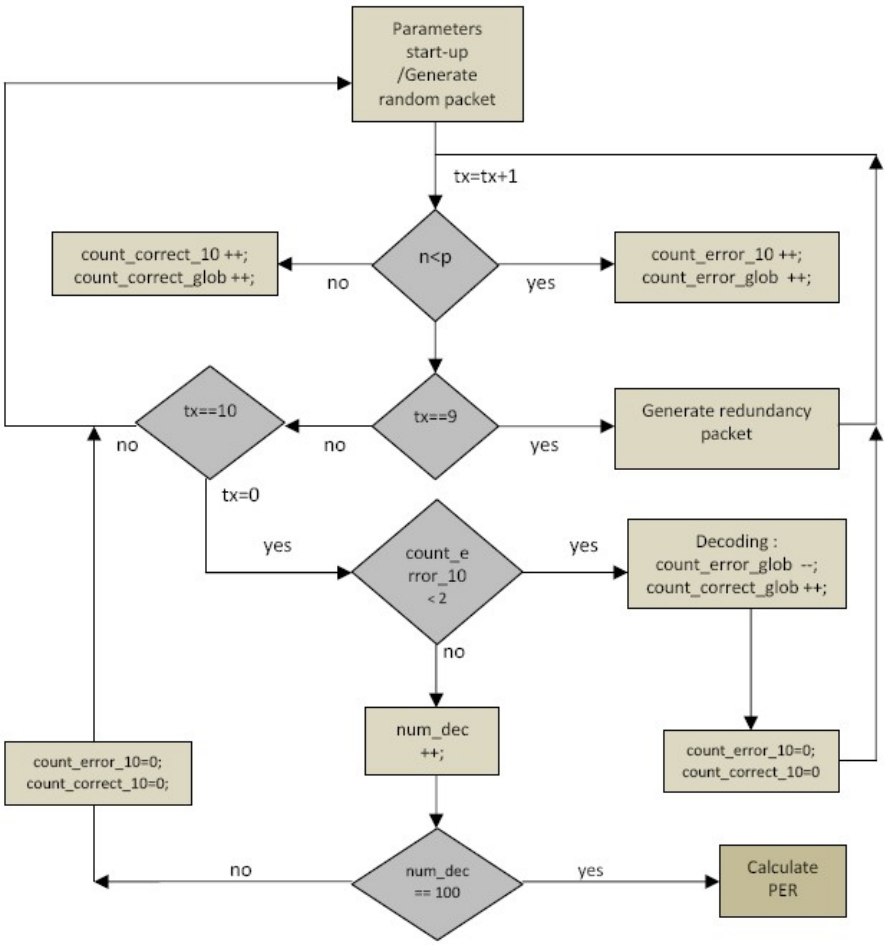

Figure 3.8: *Flow diagram of the PER simulation*

- *cont_error_10* value is 1; transmission with one packet erasure, the decoding process works successfully and after the decoding, all packets are successfully received;
- *count_error_10* value is more than 1; transmission with more than one packet erasure; In this case the variable *num_dec* that count the number of times that the decoding technique fails is increased;

When the variable *num_dec* exceed its limit, the *Packet Erasure Ratio* is calculated as ratio between the *num_dec* and the total number of source block transmissions. The results of the simulation are shown on Figure 3.9. From the graph illustrated on Figure 3.10 it is possible

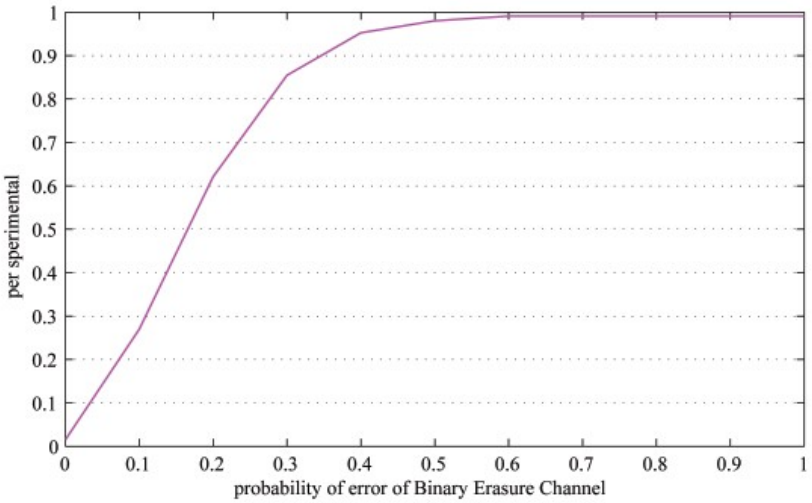

Figure 3.9: *Simulated evolution of PER using Single Check Parity Code*

to validate the theoretical expectations and verify that the results of the simulation confirm the theoretical evolution of the *Probability of Error*.

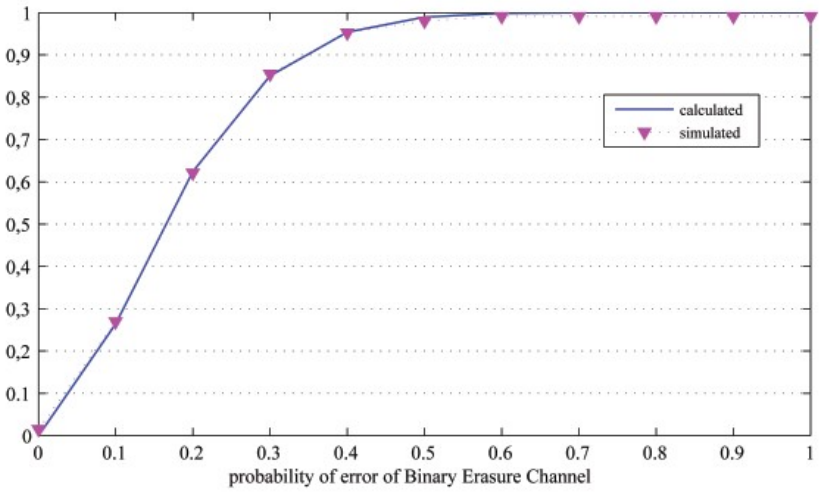

Figure 3.10: *Theoretical evolution vs. simulated results in the estimation of PER*

Chapter 4

Implementation of Packet Erasure Correcting Codes in Wireless Sensor Networks

For the experimental part of this thesis and the implementation of the packet erasure codes, it was used the development kit CC2430 from the Texas Instruments containing the necessary hardware illustrated on Figure 4.1, while for the programming of the wireless nodes it was used the IAR Embedded Workbench software environment. The kit includes all required hardware and software necessary to evaluate, demonstrate, prototype and develop several applications based on 802.15.4 network standard.

4.1 Hardware components

4.1.1 CC2430 Modules

To enable programming and networking with wireless sensors, the Texas Instruments, USA manufactures different modules, namely evaluation board (CC2430EB) and evaluation module (CC2430EM). CC2430EB includes a digital signal controller, RS-232 interface, user LEDs, user push button switches, and various other components (Figure 4.2). CC2430EM (Figure 4.3) is used for receiving and transmitting data from routers/end devices.

Figure 4.1: *Hardware components*[14]

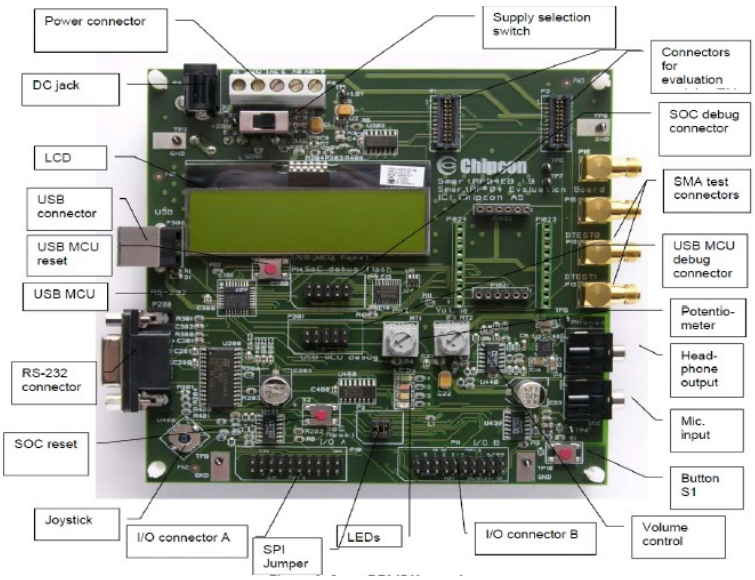

Figure 4.2: *CC2430EB evaluation board*[14]

Figure 4.3: *CC2430EM evaluation board*[14]

The basic components of CC2430EB are:

- *USB Interface:* The USB interface is used as interface to a PC and for programming and debugging using the PC debugging tools and programmers. The CC2430EB can be bus powered from the USB interface.

- *RS-232 interface:* The RS-232 can be used by custom applications for communication with other devices. The RS-232 interface utilizes a voltage translation device so that the RS-232 port is compatible with bipolar RS-232 levels. Note that this RS-232 level converter contains a charge-pump power supply that generates electric noise.

- *User interface:* The CC2430ZDK (ZigBee development kit) includes a joystick and a push button as user input devices, four LEDs and a 2x16 character LCD display as user output devices. The display and user interface is controlled by the application example program loaded in the CC2430.

4.1.2 CC2430 Chip

The CC2430 is a single-chip, IEEE 802.15.4-compliant, and $ZigBee^{TM}$ SOC (system on chip) RF transceiver with integrated microcontroller. It provides a highly integrated, flexible low-cost solution for applications using the worldwide unlicensed 2.4 GHz fraquency ISM band. The CC2430 ZDK (ZigBee development kit) is a powerful tool developing complete $ZigBee^{TM}$ applications. The hardware contains an integrated PCB antenna, the IEEE 802.15.4-compliant RF transceiver CC2430 with necessary support components, joystick, buttons, and LEDs that can be used for different purposes. The CC2430 is highly suited for systems requiring ultra-low power consumptions. This is ensured by various operating modes. Short transition times between operating modes further ensure low power consumption. The key features of CC2430 chip are as follows:

- **RF layout**
  - 2.4 GHz IEEE 802.15.4-compliant RF transceiver (CC2430 radio core);
  - Excellent receiver sensitivity and robustness to interferers;
  - Very few external components. Only a single crystal needed for mesh network systems.
- **Low power**
  - Low current consumption (RX: 27 mA, TX: 27 mA, microcontroller running at 32 MHz);
  - Only 0.5 μA current consumption in power down mode, where external interrupts or the RTC can wake up the system;
  - Only 0.3 μA current consumption in standby mode, where external interrupts can wake up the system;
  - Very fast transition times from low-power modes to active mode enables ultra-low average power consumption in low duty cycle systems;
  - Wide supply voltage range (2.0-3.6 V).
- **Microcontroller**

- High-performance and low-power 8051 microcontroller core;
- 32, 64, or 128 KB in-system programmable flash;
- 8 KB RAM, 4 KB with data retention in all power modes;
- Powerful DMA functionality;
- Watchdog timer;
- One IEEE 802.15.4 MAC timer, one general 16-bit timer, and two 8-bit timers;
- Hardware debug support.

- **Peripherals**
  - CSMA-CA hardware support;
  - Digital RSSI/LQI support;
  - Battery monitor and temperature sensor;
  - 12-bit ADC with up to eight inputs and configurable resolution;
  - AES security coprocessor;
  - Two powerful USARTs with support for several serial protocols;
  - 21 general I/O pins, 2 with 20 mA sink/source capability.

4.2 Software environment

For the programming of the wireless nodes it was used the IAR Embedded Workbench software environment illustrated in Figure 4.4. With the program code written using "C/C++" language the nodes are programmed to work as described in the application. Before the program code is debugged to the respective sensor node using USB port, it has to be compiled to prevent possible syntax errors. Every application that describes the functionality of each node, realized in IAR Embedded Workbench contains 4 files: one with extension .h in which are defined global parameters such as the channel used, the transmission power and the ID of the created wireless network and three others

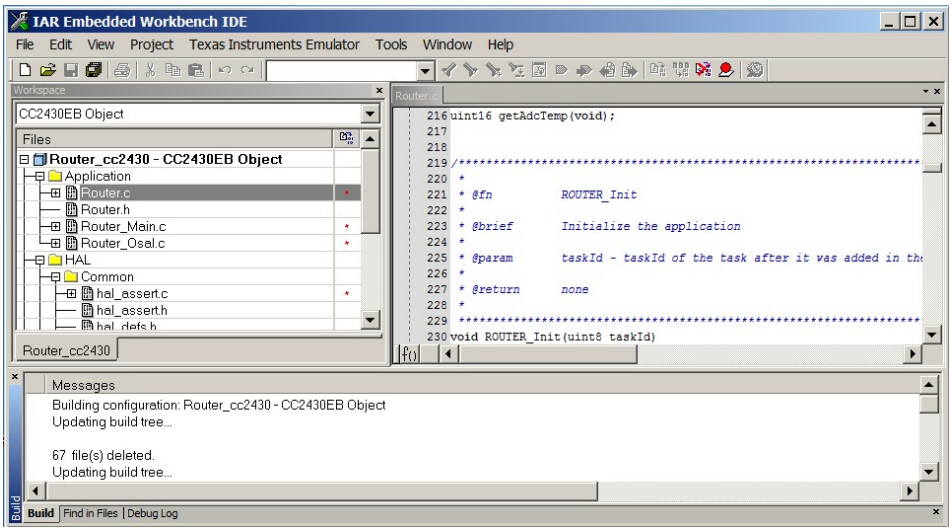

Figure 4.4: *Software environment IAR Embedded Workbench*

with extension .c. The files with extension .c contain the source code and are divided into three categories:

- File relative on the abstraction layer Operating System Abstraction Layer (OSAL), that represents an operating system serving the functionality of different components of the node;
- File containing the *Main* function, that organizes the priority of the events managed by the application;
- File containing functions like *Init*, *ProcessEvent* and *CBackEvent* and other functions necessary for the functionality of entire system application;

The last two files represent the core of the application. The *Main* module and *Init* functions initialize and start up the system. They are used to reset the MAC layer and initializes the hardware components, the OSAL and MAC layer. After the initialization of the system, the *ProcessEvent* and the *CBackEvent* manage the events occurred. The events represent facts that occur during the application running such as: the reception of a data packet, a request for association/disassociation of a node etc. These events are communicated by the MAC sublayer

to the application. The received events are controlled by the node in real time through the *ProcessEvent* and the *CBaackEvent* functionality blocks. Depending on which event was received, these two blocks respond on a well determined and specified way. That's why these two blocks are the most important part of the application. They determine the real behavior and functionality of the node.
To manage the MAC layer of every node, there are particular, specific functions that can make part of the program code, defined by libraries, so called Application Programming Interface (API). There are also other types of libraries regarding the OSAL and Hardware Abstraction Layer (HAL). The HAL libraries contain functions necessary for the activation and interaction with the timers, the LEDs present on the sensor nodes, and the display LCD present on the programming board CC2430EB (SmartRF04EB). These functions are very useful because their use can detect the occurrence of an event (blinking a LED or writing a string on the display).

4.3 TIMAC

TIMAC (Texas Instruments MAC) is a software stack that provides all libraries necessary and useful for the programming of the sensor nodes CC2430 and is certified to be compliant with the IEEE 802.15.4 standard. It includes all libraries, relevant to the MAC [21], HAL [22], and OSAL [23] layer.
The MAC library permits to program the behavior and functionality of the nodes through "C/C++" program language, maintaining high level of abstraction, without analyzing in specific the PHY and MAC sublayer. This simplifies a lot the programming process of the nodes. It describes the application programming interface for the 802.15.4 MAC software. The API provides an interface to the management and data services of the 802.15.4 stack. The other two libraries handle on simple way the hardware and the operating system of the nodes. The software stack also contains examples presenting and explaining the basic functionality of a wireless networks. The most important are:

- Nomination of a PAN node in the network;
- The association on each node to the PAN coordinator;

- The transmission/reception of data packets.

These three basic scenarios are discussed in detail in the following sections.

4.3.1 Nomination of a PAN node in a non-beacon enabled network

This scenario shows a PAN coordinator device starting a non-beacon enabled network. It first resets the MAC on device startup and performs an energy detect scan to find an unused channel. Then it performs an active scan to find the channel with the lowest number of active networks. After the active scan, sets the MAC attributes it needs to start a network: the short address, beacon payload, and associate permit flag. Then it starts a non-beacon-enabled network. Figure 4.5 illustrates the flowing diagram: A node that is nominated as a PAN coordinator of the network, has to be the first node of the network activated. To ensure this, the node after the reset of the MAC sublayer, begins an energy scan on one or more channels using the function MAC_*MlmeScanReq()* that requires from the MAC sublayer to verify which of the scanned channels are available. When the scan operation is complete, the MAC sends an event called *MAC _MLME_SCAN_CNF* to the application that contains the result of the scan operation. If the scanned channel is available, the result is positive because the node that requires the scan operation, was the first node activated in the network. The PAN coordinator in this case, sets some global parameters such as the ID of the network (PAN_ID), the value of the logical channel available (MAC_CHAN_XX), the short address etc. Then using the function MAC_*MlmeSetReq()* communicates to the MAC sublayer this parameters and notifies that wants to initialize a network through the MAC_*MlmeStartReq()* function. The MAC sublayer responds with the event *MAC_MLME _START_CNF* and if the response is positive, the node is successfully nominated as a PAN coordinator.

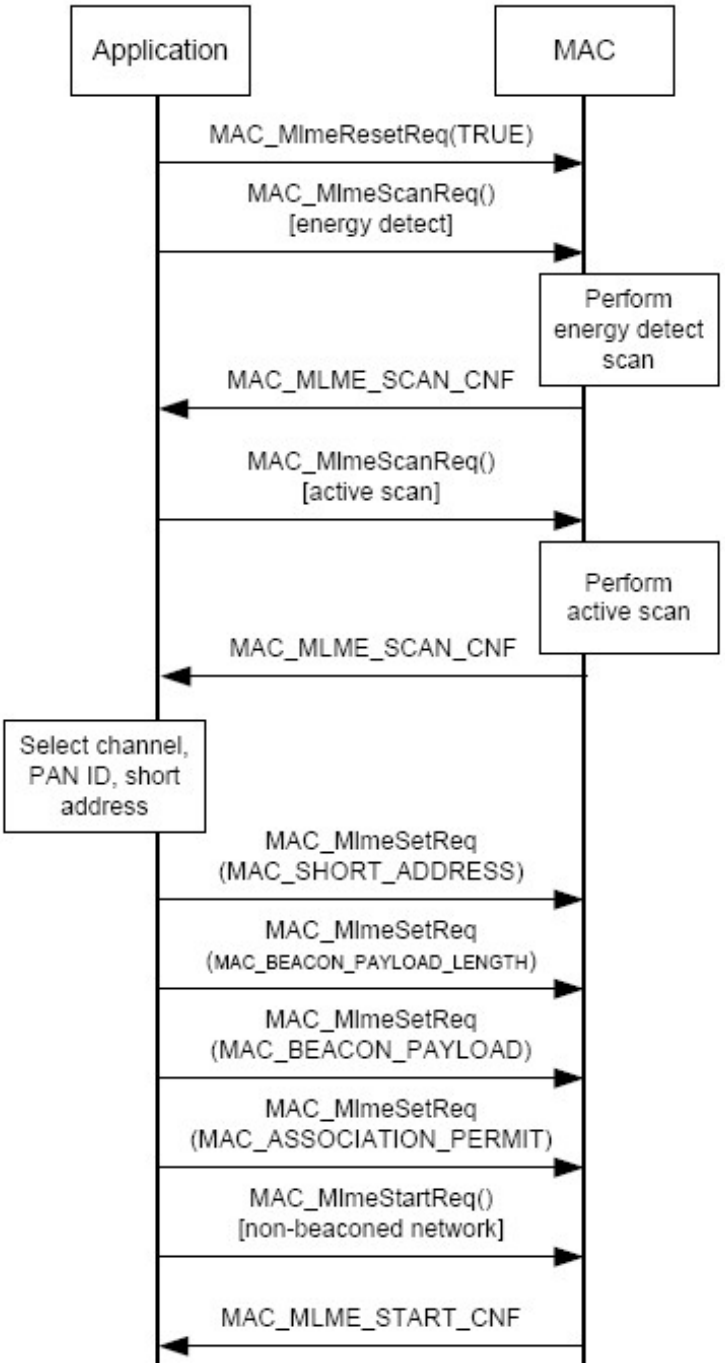

Figure 4.5: *Non beacon-enabled network start*[21]

4.3.2 Non beacon-enabled network Scan and Association

This scenario shows a device connecting to a non beacon-enabled network. Before the devices communicate between them, they have to be associated to the PAN coordinator. The device performs an active scan, broadcasting a beacon request on each channel. When the coordinator receives the beacon request it sends a beacon. When the scan is complete the MAC sends a *MAC_MLME_SCAN_CNF* with the PAN descriptors it has received during the scan. The device application examines the PAN descriptors, selects a coordinator and responds to the MAC sublayer with MAC_*MlmeAssociateReq()*. The coordinator application receives a *MAC_MLME_ASSOCIATE_IND* and calls MAC_MlmeAssociateRsp() allowing the device to associate. At this point, the MAC sublayer sends two events: a *MAC_MLME_ASSOCIATE_CNF*, that receives the device application indicating success and the *MAC MLME_COMM_STATUS_IND* which the PAN coordinator receives, indicating the result of the associate operation. The device application then sets the MAC short address attribute.
A possible negative response from the PAN coordinator on a association request could be an exceed of the maximum number of devices that can make part of the network. The maximum number of devices is introduced to prevent a possible network traffic congestion.

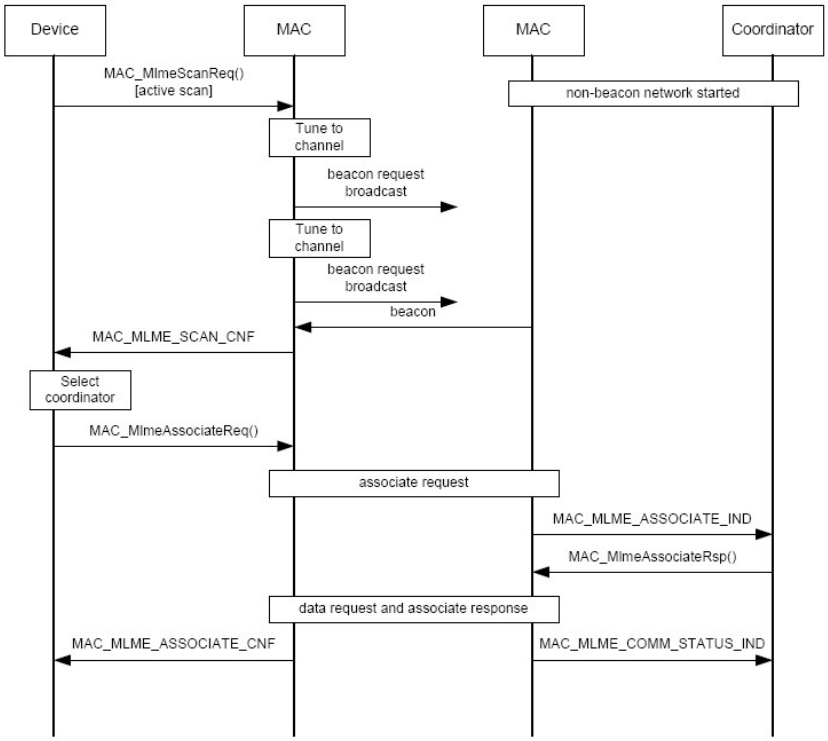

Figure 4.6: *Device connecting to a non beacon-enabled network*[21]

4.3.3 Data transactions (Transmission and reception of data packets)

This scenario shows various direct data transactions between a Full Function Device (FFD) device and a coordinator. A basic data transaction is as follows: The device application calls MAC_*McpsDataReq()* function, to indicate that it wants to send a data frame. The MAC transmits this frame and receives an acknowledgement. The MAC sends to the device application a *MAC_MCPS_DATA_CNF* with status indicating success. On the receiving side, the MAC sends the coordinator application a *MAC_MCPS_DATA_IND* containing the received data frame. Normally, in a transmission of data packets, the

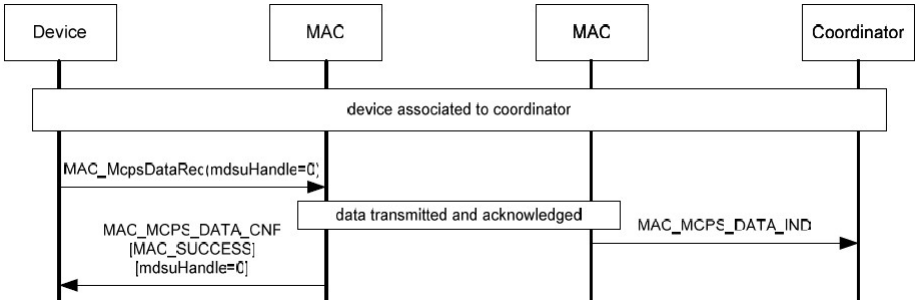

Figure 4.7: *Data transactions*[21]

packet contains only the address of the destination node. When the network uses flooding algorithm to route the packet to its destination, all nodes not interested in that packet simply ignore the packet as the destination address does not coincide with their own address. For broadcast transmissions, it is necessary to specify a destination address of the packet the special value 0xFFFF. A packet data that is transmitted via broadcast is addressed to all nodes of the network, but only the nearest nodes to the source node will receive the packet. In the developed applications, these events (the transmission and the reception of the packets), are the parts of interest where the implementation of the encoding and decoding algorithms takes place.

4.4 Packet Sniffer

The Packet Sniffer is a software produced by Texas Instruments. It is a powerful instrument especially used in the development phase of an application. In this thesis was very useful because it can capture, filter and decode IEEE 802.15.4 MAC packets, and displays them in a convenient way and in a real time, following the chronological order of the transmission of the packets. On this way, it was possible to display the content of the packet and the application data and verify that the redundancy packets were constructed exactly as was required in the program code. Figure 4.8 shows the packet sniffer environment. The packet sniffer requires a single CC2430EB with a CC2430EM con-

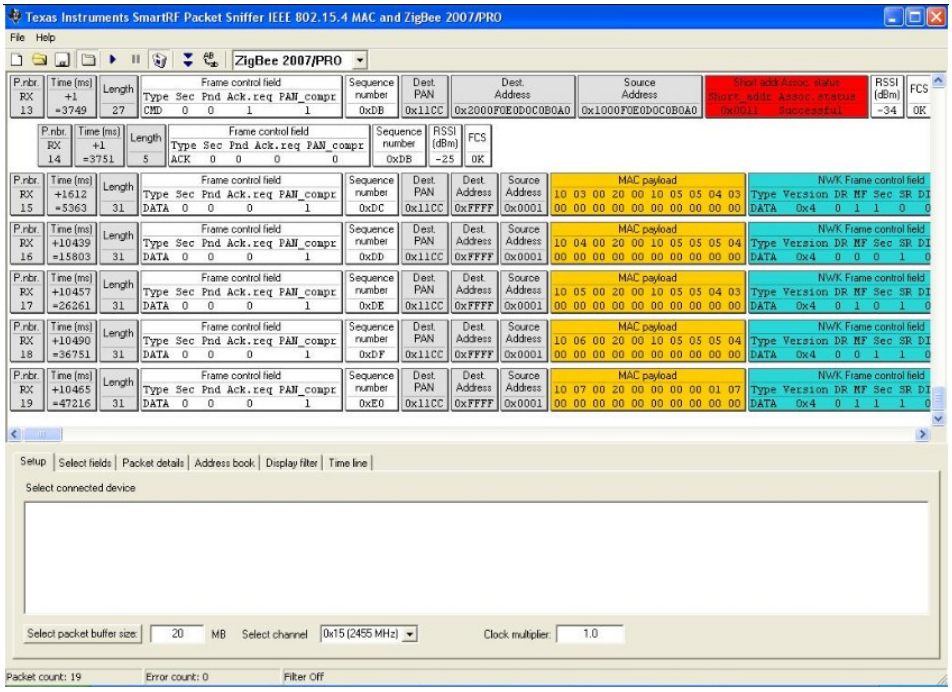

Figure 4.8: *Packet Sniffer screenshot*

necter to it. The CC2430EB board is connected to the PC through a USB cable. This board is dedicated only to listen the specific channel used in the application and the number of the channel is selected by the software that controls Packet Sniffer (in the applications de-

veloped, it was used channel number 21). During the formation of the network the connected board displays the transmitted packets in temporal chronological order. For each packet it is displayed useful information such as: the destination address of the packet, the source address, the type of the packet (data packet, ACK o command packet) and the application data. With this software tool it is possible understand if all nodes are associated with the PAN coordinator, which node transmits data packets etc. In a development phase, this instrument is very useful because it is possible also to identify the nodes that are communicating in the network, the ID of the send packets and understand the real time functionality of the entire network.

4.5 Developed Applications

In this thesis, with different experimentation tests, it has been attempt to prevent the retransmission of an eventually erased packets during the transmission. Instead of using the ARQ protocol, two FEC codes have been implemented for the recovery of the erased packets, . In this way, the FEC algorithm by performing the recovery of the packet losses at the receiver, prevents their retransmission. According to the variation of the transmission power, the distance of the nodes and the topology of the network, it was evaluated the convenience of introducing an encoding/decoding algorithms in the transmission chain.
The implementation of the FEC algorithms considered two different types of codes:

- Single Check Parity Code

- Hamming Code

In both applications were used the following global parameters:

- Through the sixteen different transmission channels present in the 2.4 GHz band it was chosen the 21^{th} channel;

- For the ID of the network (PAN ID) it was chosen 0x11CC;

- It was chosen the mesh network topology. Each node is initialized as an FFD, in this way the other nodes that want to associate could be added by any other device already associated with the network;

- Each node has two different addresses: a short address of 16 bits and an extended address of 64 bits. In this way each node is uniquely determined in the network. These addresses are assigned in the initialization phase depending on how the nodes are programmed. For the PAN coordinator, both addresses are determined in the initialization phase of the application. For the others FFD devices only the extended address is determined in the initialization, the short address instead is assigned by the PAN coordinator during the association phase. In these applications the following addresses were chosen:

  Extended address of the PAN coordinator:
  {0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88};

  Short address of the PAN coordinator:
  {0xAA, 0xBB};

  Extended address of the FFD devices. The last 2 bytes vary from 0x10 to 0xF0:
  {0xA0, 0xB0, 0xC0, 0xD0, 0XE0, 0xF0, 0x00, 0xXX}

  Short addresses of the FFD devices are assigned by the coordinator to which the node decides to associate.

4.6 Implementation of FEC algorithm using Single Check Parity Code

The first developed application regards the implementation of an encoder that adds a redundancy parity packet calculated as a function of the data packets. It was used a single check parity code (9,10) i.e. the sender transmits 9 data packets and the 10 th is calculated as a parity packet using a vertically xor-operation between the other 9 data packets. In this section are explained the functions used for the generation of the packets and for the realization of the encoder

and decoder algorithms. Figure 4.9 illustrates the encoding and the decoding process.

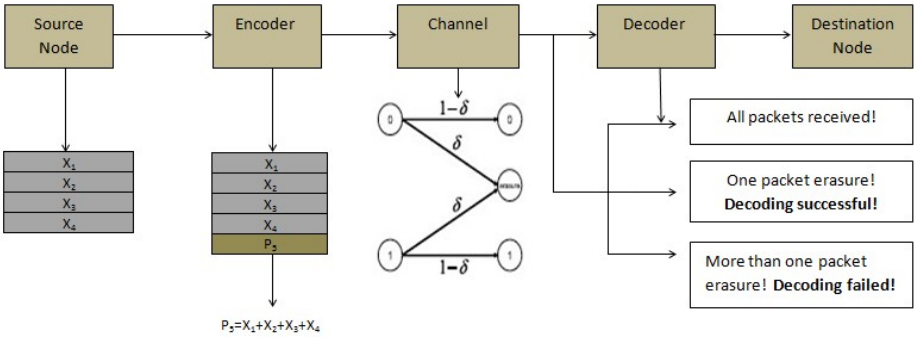

Figure 4.9: *The transmission chain using Single Check Parity Code*

For the generation of the packets it was implemented a function called *Data_create()*. *Data_create()* generates application data packets containing 20 bytes. Each packet is composed by the *Router_Packet_Header* part and the *Router_Packet_Data* part. The header contains information useful for the encoding/decoding algorithms (the *SBN_ESI*, explained better bellow), and for the routing of the packet (the destination address of the packet and the source address of the sender node). The application data in this case, contain random values. The 10^{th} packet is the redundancy packet, calculated as a parity packet of the previous data packets. Figure 4.10 illustrates the contents of a single packet. Before the sender node transmits the data packet, is memo-

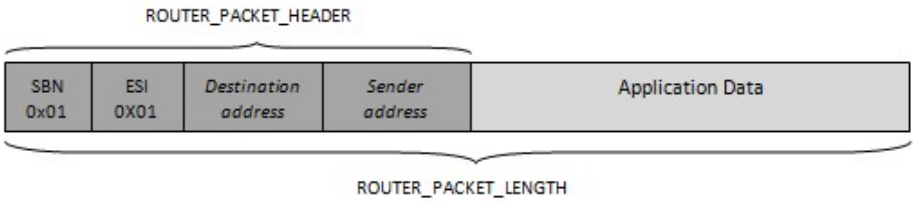

Figure 4.10: *The packet generated by the Data_create() function*

rized in a source block (matrix structure called *Router_CopyData*) in a well determined order, considering the first 2 bytes of each packet

called *SBN_ESI*. The first byte is the *Source Block Number* indicating the ID of the source block of which the packet belongs in the encoding process, and the second one is *Encoding Symbol ID*, indicating the row in the Source Block. The encoding process and the source block are illustrated in Figure 4.11.

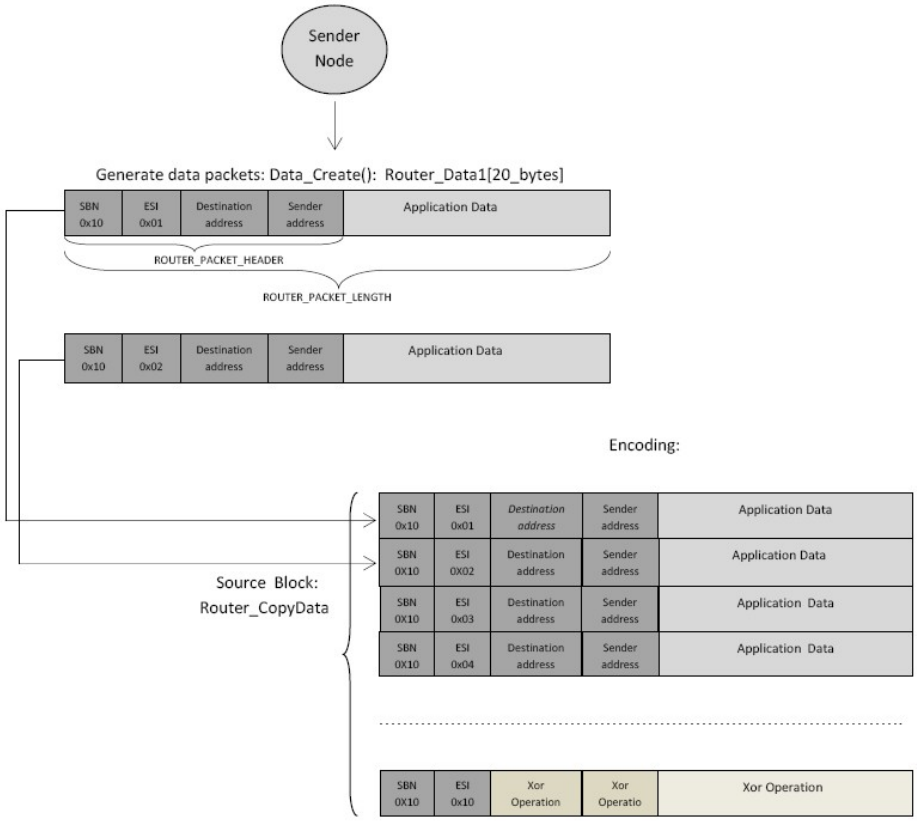

Figure 4.11: *Encoding technique*

In order to send the generated packet, the sender node calls the *MAC_McpsDataReq()* function that sends the application data to the MAC sublayer for transmission in a MAC data frame. The application sets data pointer *msdu.p* to point to a buffer containing the data the application is sending. Figure 4.12 shows how a buffer containing the data can be constructed. Another function called *MAC_McpsDataAlloc()* can be used to simplify allocation and preparation of the data buffer. This function allocates a buffer of the correct size to contain the parameters, MAC header, and application data and prepares it as described in Figure 4.12.

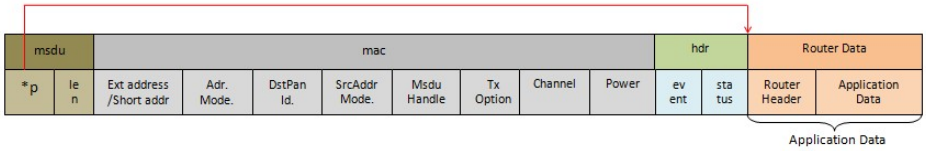

Figure 4.12: *MAC data frame containing the application data*

The MAC sublayer responds to MAC*McpsDataReq()* function with the event *MAC_MCPS_DATA_IND* that sends data from the MAC sublayer to the application. The parameters for this event points to a dynamically allocated memory buffer. When the MAC allocates a buffer for the received data it can allocate extra space in the beginning of the buffer for application-defined data. The structure that arrives at the receiver is the following:

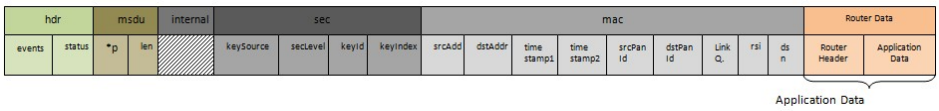

Figure 4.13: *MAC data frame arrived at the destination node*

To the sender node, the MAC sublayer sends the event called *MAC_MCPS_DATA_CNF* every time *MAC_McpsDataReq()* is called. The event returns the status of the data request. The possible status could be: *MAC_SUCCESS* (operation successful), *MAC_CHANNEL _ACCESS_FAILURE* (data transmission failed because of the congestion on the channel), *MAC_NO_ACK* (no acknowledgment was received from the peer device) etc.

The destination node, after the reception of each packet, detects its *SBN_ESI* and uses a *flag_array* to memorize the reception of that packet. A function called *ArrayFlag()* is used to notify the reception. For example, if the received packet was the packet with ID(EBN)=0x01, it sets a flag on the first position of the *flag_array*, corresponding to the ID of the received packet. Then it stores the entire packet in the decoding buffer (matrix structure called *Router_MatrixRx*). When a packet that belongs to another source block (in this case SBN=0x11) arrives at the receiver, it calls a specific *Counter_Ones()* function to determinate the number of received packets with the same *Source Block Number* (SBN=0x10). A variable named *counter* runs over the *flag_array* and counts the number of received packets. The decoding process is illustrated in Figure 4.14. According to the value of counter,

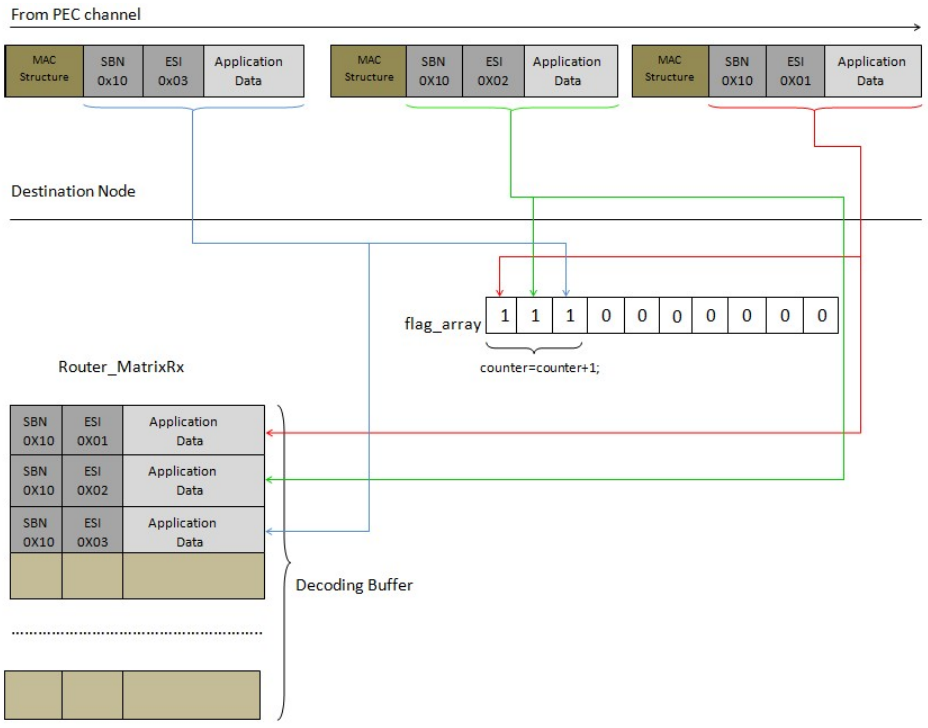

Figure 4.14: *The decoding used in FEC Single Check Parity Code algorithm*

three different situations of interest were addressed, namely:

- Transmission without packet erasures, all packets received;
- Transmission with one packet erasure;
- Transmission with more than one packet erasure.

Transmission without packet erasures

If all packets are received correctly at the destination node, the decoding process is unnecessary. In this specific case, the variable *counter* assumes value 10, or 9 in case the redundancy packet is the lost packet. This packet added by the encoder is not useful at the receiver.

Transmission with one packet erasure

In case of one packet erasure, the decoding process works successfully, and it can recover any packet erasure simply applying a xor-operation between the other received packets containing the same *Source Block Number*. An example is illustrated on Figure 4.15. For simplicity, there are four data packets and the fifth packet is a redundancy one. In this example, the second packet has not been delivered at the receiver, which extracts its ID and applies the decoding process to recover it. The decoding in this case works independently of the ID of the erasure packet.

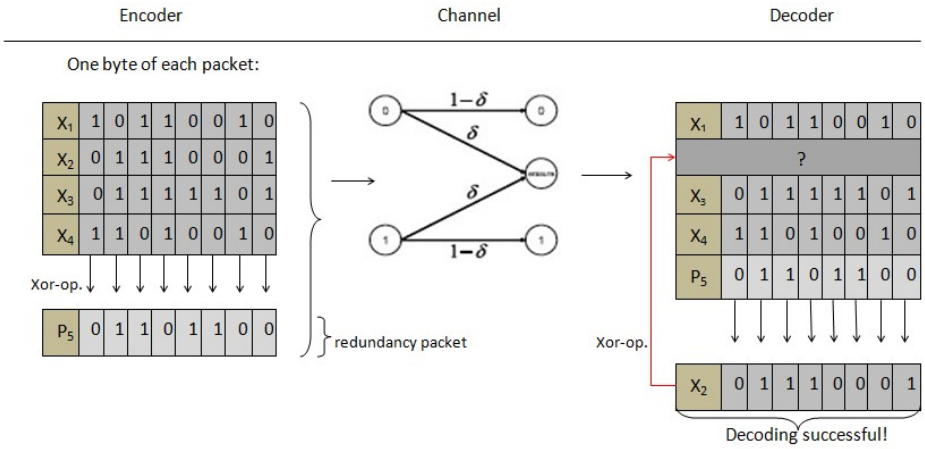

Figure 4.15: *Successful decoding in FEC Single Check Parity code algorithm*

Transmission with multiple packet erasures

In this case the decoding process fails and the receiver isn't capable to recover the packet losses.
The introduction of the parity code is worth in case the channel is characterized by low noise and as consequence perturbs the transmission only with one packet erasure. In some cases the application of this code may be worth, while in some others is better the use of another code more powerful codes. An ARQ protocol, may be preferable, depending of the propagation environment, the application, the distance of the nodes and the topology of the network.

4.7 Implementation of FEC algorithm using Hamming(7,4) code

The second developed application regards the implementation of an encoding/decoding algorithm that uses the Hamming code (7,4). In this case, the encoder adds 3 redundancy packets to four data packets, calculated considering the Hamming generation matrix i.e., the

following parity equations:

$$P_5 = X_1 + X_2 + X_3$$

$$P_6 = X_1 + X_2 + X_4$$

$$P_7 = X_1 + X_3 + X_4$$

The decoding process at the destination node is capable to recover one or two packet erasures, one more, compared to the previous application in which the single parity code is used, but with the essential difference that the parity code introduces only one redundancy packet and the Hamming code three packets every four ones. It's obvious that the implementation of encoding/decoding algorithms based on parity code requires less overhead information than encoding/decoding algorithms using a Hamming code. Figure 4.16 summarizes the encoding and decoding processes:

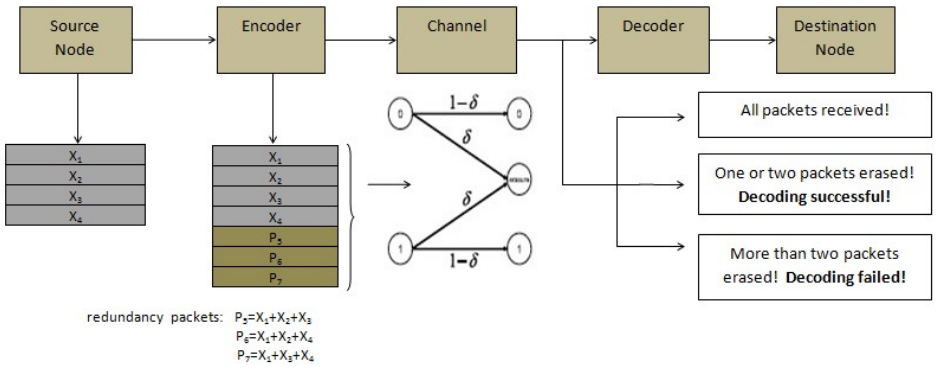

Figure 4.16: *The transmission chain using Hamming (7,4) Code*

The sender is programmed to trigged every 5 seconds an event that calls the *Data_create()* function, thus, generates a data packet and stores it in the Source Block (a matrix structure called *Router_CopyData*) in a well determined order, according to its IDs. As in the case where it was used the parity code, in this case also, the packets are memorized in the source block considering the first two bytes of each packet *(SBN_ESI)*, that identify the ID of the packet and the *Source Block Number* to which it belongs. After the transmission of four data packets, the sender calculates the redundancy packets and sends them to

the same destination.
The decoding process at the receiver is triggered with the arrival of packet containing a different *Source Block Number* different from that of the packets in the current source block. The reception of a single packet is notify by adding a flag in the opposite *flag_array* at a well determined position, according to the ID (EBN) of the received packet. The variable named *counter* updates its value every time a packet arrives at the receiver.In this case however, the number of correctly received packets is not sufficient; it is necessary also the ID of the erasure packets. With this purpose three different arrays have been defined: the first one mentioned above, called *flag_array* that notifies the arrival of a specific packet by setting a flag on a well determined position, the second one called *array_id_miss* containing the IDs of the erased packets, and the third one called *array_id_not_miss* containing the IDs of the correctly received packets. All these arrays contain information relevant to the data packets. Regarding the redundancy packets, another two additional arrays were defined, that have the same goal as the *array_id_miss* and the *array_id_not_miss*, memorizing the IDs od the redundancy erasure packets (*array_id_miss_red*) and those received correctly (*array_id_miss_not_red*). Figure 4.17 illustrates the use of these arrays: For the memorization of the IDs of

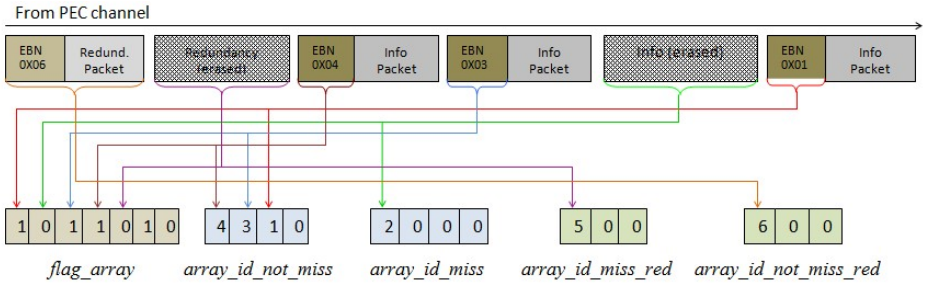

Figure 4.17: *The use of the arrays in the FEC Hamming code algorithm*

the data packets, the function *Save_Id()* was implemented that runs the *flag_array* and depending of the presence or not of the flag in the array, memorizes the IDs of the packets in the arrays mentioned above. For the redundancy packets instead, it was implemented *Save_Id_Red()*

that behaves exactly on the same way as the *Save_Id()* working on the redundancy packets and not on the data packets.
It were implemented two different functions to separate the case in which data packets are missing from the case in which redundancy packets are missing. In the first case, the decoding process has to be trigged for the recovery of the missing data packets since this is the case of interest, otherwise the decoding isn't necessary. The decoding process regards three different situations:

- Transmission without packet erasure;
- Transmission with one or two packet erasures;
- Transmission with more than two packet erasures;

Transmission without packet erasure

In this situation the value of the *counter* is equal to the total number of data packets and the decoding process isn't necessary. The addition of redundancy packets influences only the bit rate of the channel.

Transmission with one or two packet erasures

The transmission affected by one or two packet erasures triggers the decoding process that attempts to recover the data packets applying the parity equations from which is possible obtain the missing packets. Given that the minimum distance of the Hamming code (7,4) is d_{min}=3, is possible to repair $d_{min} - 1 = 2$ packets with probability 1. The packets that have been erased could be either or data packets or redundancy packets. The situations of interest are those where at least one data packet has been erased. Figure 4.18 shows transmission affected by two packet erasures, one redundancy packet and the other that is a data packet. It is evident that the system of parity equations contain in this case two equations of interest through which is possible recover the lost packet. Such equations are:

$$X_1 = X_2 + X_3 + P_5$$

$$X_1 = X_3 + X_4 + P_7$$

Applying the equation $X_1 = X_3 + X_4 + P_7$, the data packet can be recover simply through xor-operation. Figure 4.19 shows an example

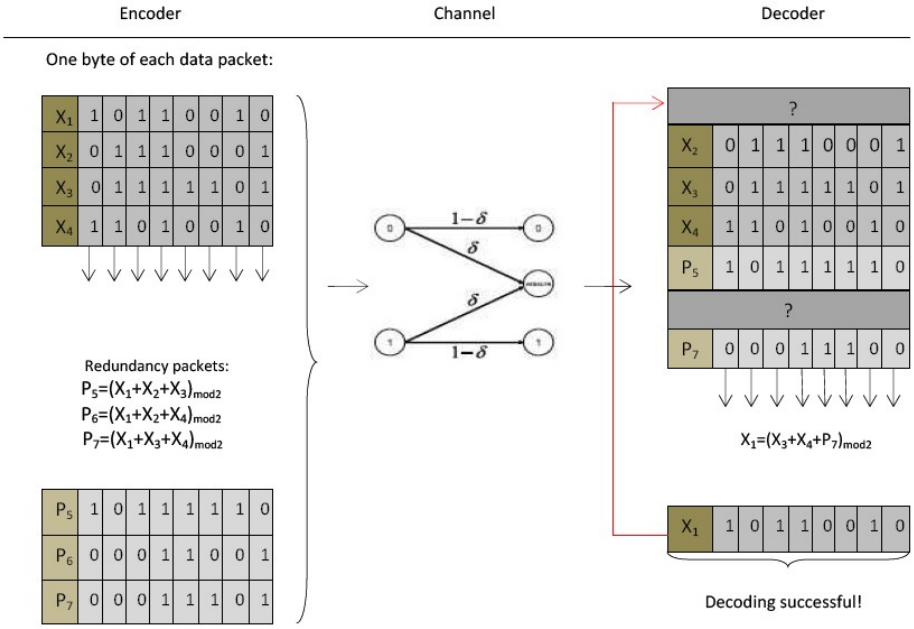

Figure 4.18: *Example of transmission with two packets erased*

in which the transmission was affected by two data erasures. In this case the system of parity equations requires a solution by substitution of equations. It is necessary to recover at first one of the erased packets applying an equation that does not contain lost packet, i.e., (X_1=X_3+X_4+P_7) than using its recovery, it is possible to recover also the second one.

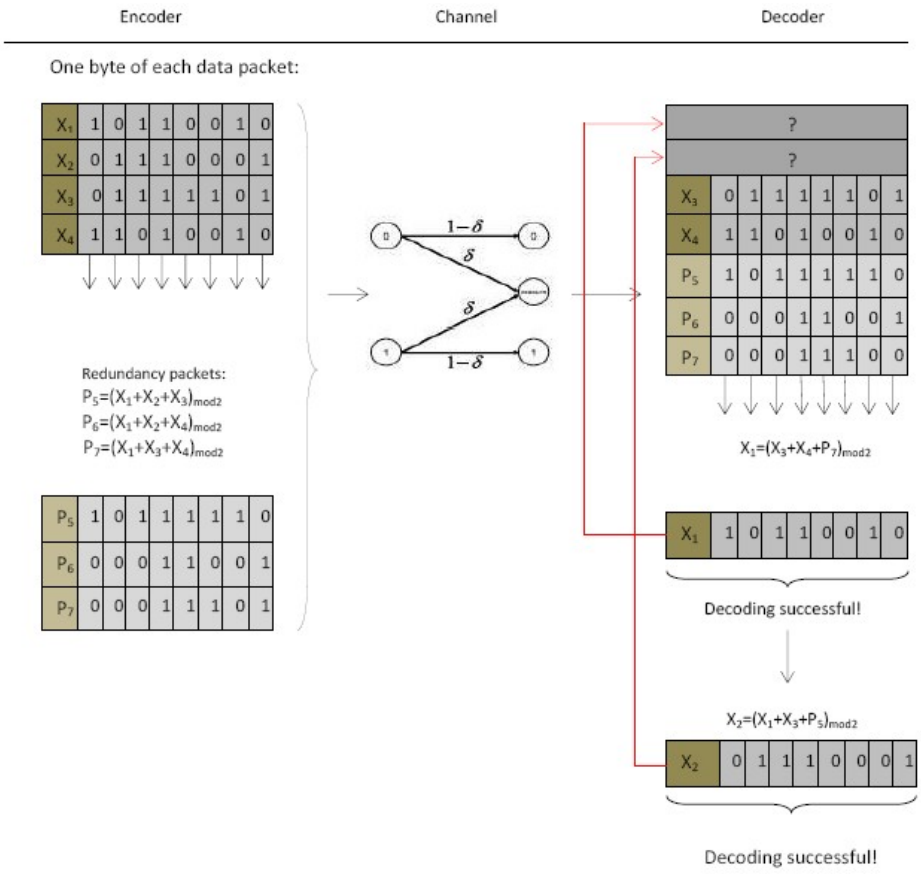

Figure 4.19: *Example of transmission with two data packets erased*

4.8 Implementation of ARQ protocol

In order to evaluate the advantage of the implementation of the FEC algorithms explained above , it was necessary to implement another scenario that makes use of the ARQ protocol. The ARQ protocol considers retransmission of every not-acknowledged packet by the destination node. In that way it is possible make a comparison and understand the benefits of each single implementation. Figure 4.20 illustrates an example. In the implementation of this scenario, the source node has

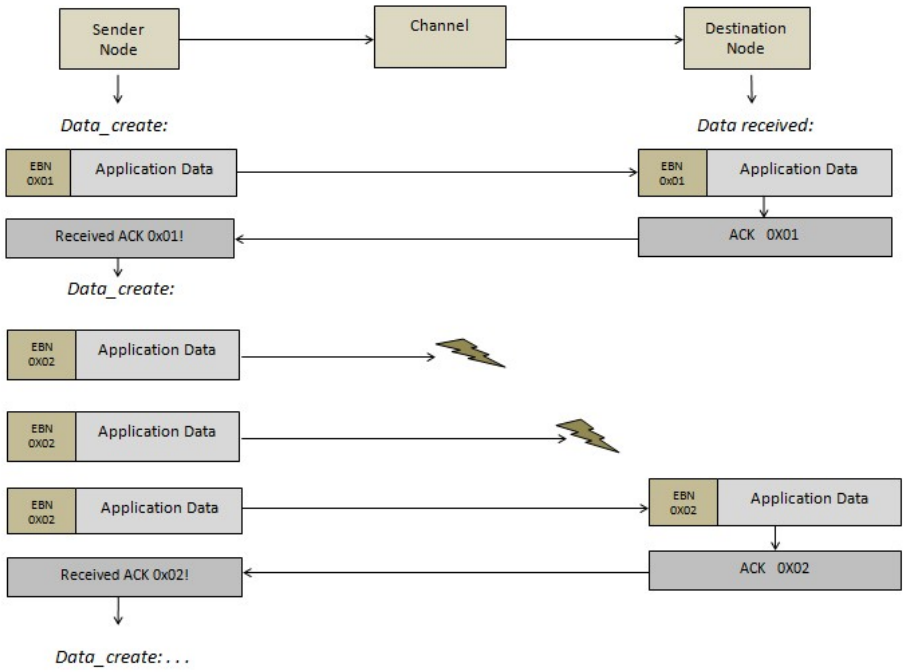

Figure 4.20: *Example of transmission using ARQ protocol*

to be enabled to retransmit the not-acknowledged packets. To ensure this, the *txOption* filed making part of the *Router_McpsDataReq()* function has to be set with the *MAC_TXOPTION_ACK* attribute. Every time it sends a packet, the MAC sublayer responds with an event called *MAC_MCPS_DATA_CNF*, containing the result of the transmission. In case the transmission was successfully completed, the sender

node continues with the transmission of the following packets, otherwise it retransmits the packet again.

In order to evaluate the convenience of each scenario explained above, in the experimentation part three different cases were considered. By making a comparison between the scenario using the FEC algorithm, another one that uses the ARQ algorithm and the last one that ignores packet erasures, it is possible evaluate the number of correctly received data packets at the receiver and verify the advantage/disadvantage that each scenario implies.

Chapter 5

Results and conclusions

This Chapter presents the results and the conclusions reached during the various simulations and field trials. In order to confirm the correctness in the realization and the implementation of the encoder/decoder, it was realized a simulation of the transmission chain, considering two different cases: the first one regards the *Single Check Parity Code* and the second one the *Hamming(7,4) Code*. In both cases, it was evaluated the *Probability of Decoding Failure* through the PER parameter. The various simulations consist in transmitting a source blocks of data and redundancy packets through the PEC channel. The channel is simulated via software, using the *Monte Carlo* method. Varying the probability of loss of the channel, it was evaluated the ratio between the number of failed decodings and the number of transmitted source block packets. Through this simulation it was possible validate the theoretical evolution of the probability of decoding failure and the correct functionality of the coding/decoding technique.

On the other hand, the experimentation tests, consist of the realization of two different propagation scenarios. The first one is realized without any presence of obstacles on the main propagation path between the nodes, while the second one addresses a situation at which the main propagation path is obstructed by the presence of obstacles. In these scenarios it was considered the multi-hop topology of network. In order to evaluate the convenience of using the FEC algorithms instead of an ARQ protocol, it was evaluated the number of correctly received data packets in front of a fixed number of transmit-

ted data packets in both cases. Through the evaluation of the Packet Loss Ratio (PLR) in different packet recovery schemes, it is possible evaluate the convenience of introducing the recovery packet techniques described above.

5.1 Results of the simulation using a $(10, 9)$ Single Check Parity Code

The first scenario regards a network composed of three sensor nodes: a node that represent the PAN coordinator of the network and two nodes (the source node and the destination node) that after the association phase to the PAN coordinator initialize the transmission. Figure 5.1 illustrates the created network. If p is the probability of loss for a

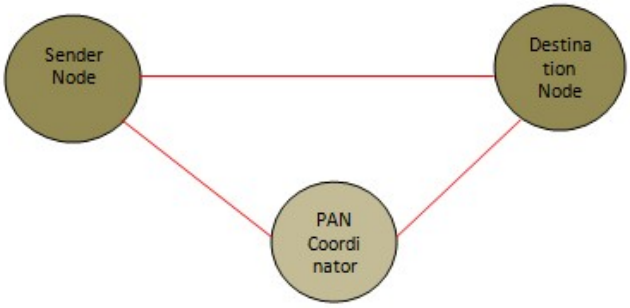

Figure 5.1: *The network created*

generic single packet through the PEC, and if erasures occur independently of each other then the probability of decoding failure is:

$$\begin{aligned} P_{df} = P[\text{uncorrectable error}] &= P[\text{2 or more packet erasures}] \\ &= 1 - P[\text{1 or less packet erasures}] \\ &= 1 - P[\text{no erasures}] - P[\text{1 erasure}] \\ &= 1 - (1-p)^{10} - 10(1-p)^9 p \end{aligned}$$

The expected value of the simulation parameter *Packet Erasure Ratio* (PER) is the *Probability of decoding failure*. The PER is calculated after the variable that counts the number of failed decoding exceed

its limit (100 failed decodings). The results of the simulation and the comparison with the theoretical evolution are presented on Figure 5.2. The blue line represents the theoretical evaluation of the *Probability*

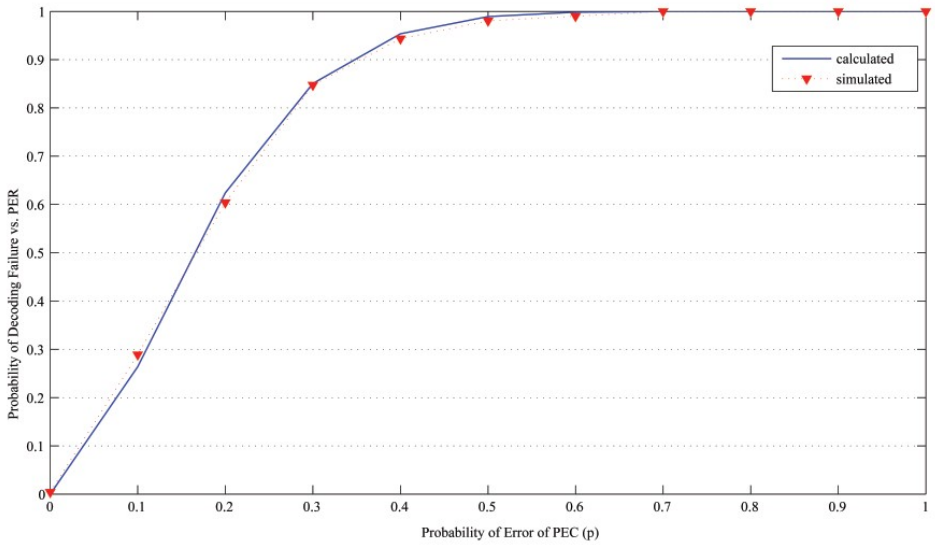

Figure 5.2: *Simulated vs. Theoretical evolution of the Probability of Decoding Failure using Single Check Parity Code*

of Decoding Failure and the red triangles the simulated values. These values as shown, follow the evolution of the P_{df}.

5.2 Results of the simulation using Hamming (7,4) Code

In case the Hamming Code is used, the transmission consists of source blocks containing 7 packets through the PEC channel. The last three packets, as mentioned, are redundancy packets calculated using the generator matrix $G_{hamming}$. The decoder at the receiver is capable, in this case to recover one or two packets erasures. If p is the probability of loss for a generic single packet over a PEC, than the *Probability of*

Decoding Failure is:

$$P_{df} = P[\text{uncorrectable error}] = \sum_{t=3}^{7} \binom{7}{t} p^t (1-p)^{7-t}$$

The comparison between the experimentation results and the theoretical evolution of the *Probability of Decoding Failure* are illustrated on Figure 5.3. The comparison of the two graphics is presented in Figure

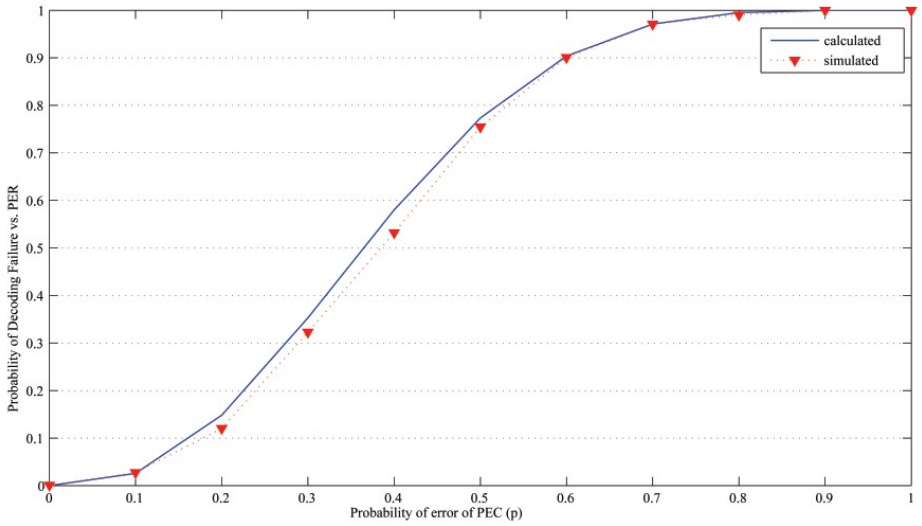

Figure 5.3: *Simulated vs. Theoretical evolution of the Probability of Decoding Failure using Hamming(7,4) Code*

5.4.

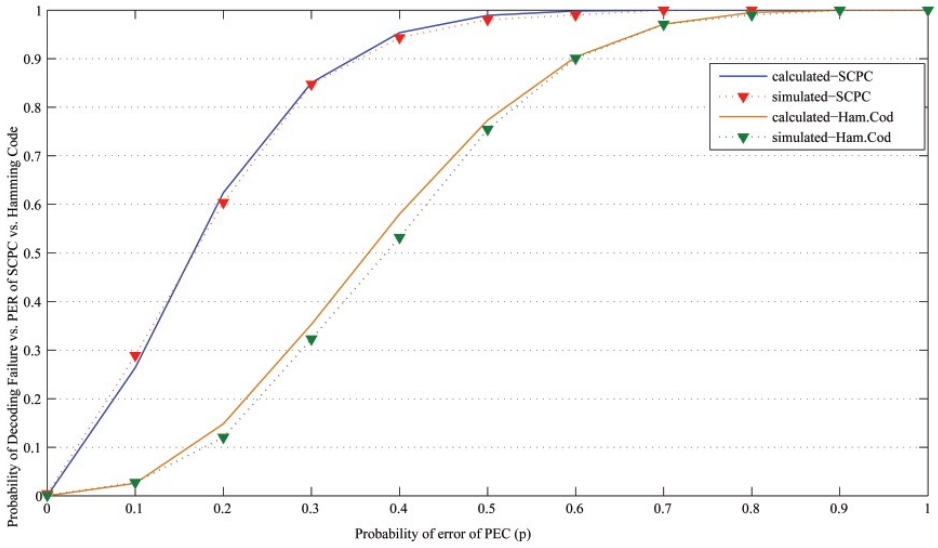

Figure 5.4: *Confront of the simulated values and the theoretical evolution of the Probability of Decoding Failure*

The simulated values of the PER parameter compared to theoretical evolution of the *Probability od Decoding Failure* confirm the correctness in the implementation of the recovery techniques introduced.

5.3 Experimental Activity

The experimental part consist in evaluating the convenience of introducing the FEC techniques in the transmission chain. The core idea to propose this technique is, as it was mentioned before, the necessity to reduce as much as is possible the energy consumed in the entire WSN. It ware considerate three cases:

- WSN using the FEC techniques for the recovery of the packet erasures;
- WSN using the ARQ protocol for the recovery of the packet erasures;
- WSN in which the packet erasures are ignored.

The considered network is illustrated on Figure 5.5. In order to

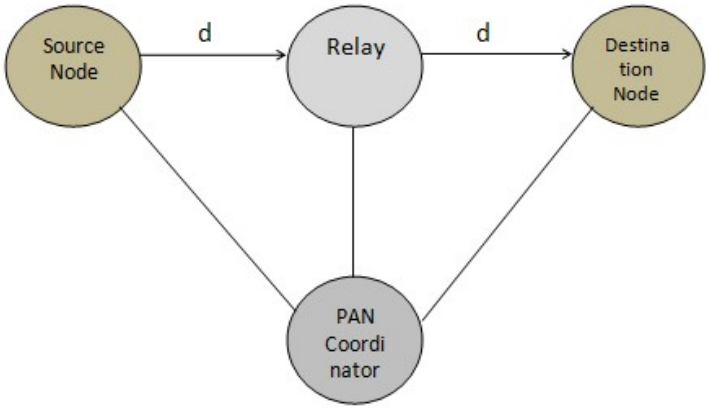

Figure 5.5: *The network realized for the experimentation*

distinguish the rules of the nodes in the network, it was used the last byte of the extended address *Router_ExtAddr2*, a variable called ROUTER_LAST-_EXT_ADDR. Setting its value to 0x10, 0x30 or 0x20 it was possible distinguish the source node, intermediate node (relay) and the destination node in the network. In the following sections is presented a brief description of the three cases.

5.3.1 Using the FEC technique for recovery of the packet erasures

In this scenario, the **source node** is programmed to generate packets every 5 seconds using the *Data_create()* function. To avoid the retransmission of the not acknowledged packets it is necessary modify into the *Router_McpsDataReq* function the *txOptions* filed setting the MAC_TXOPTIONS_NO_RETRANS attribute. The short address of the destination node (in this case the intermediate node) is specified as input argument in the *Router_McpsDataReq* function. Every time the source node generate and send a data packet, update the value of a variable called *counter_tx*. The results are taken when this variable exceed its limit (100 data packets send). In this case, has to be taken into account the fact that redundancy information also

has to be transmitted. The **relay** instead, should act as receiver and transmitter. Normally, it should receive the arriving packets from the source node, memorize them, modify the destination address and send them to the destination node. The arrival of a packet belonging to the next source block triggers the decoding algorithm through the link between the source node-relay and send them also to the destination node. Thus, the operations that has to be made by this node are more complicated. The **destination node**, every time that receives a data packet increments a variable called *counter_rx*. Eventual erased packets through the link relay-destination node could be recovered, applying also the decoding technique at the destination node.

5.3.2 Using the ARQ technique for the recovery of the packet erasures

In the realization of this scenario, the **source node** has to be able to retransmit the generated packet every time it don't receive the ACK from the intermediate node. The retransmission of the packet is possible by setting the attribute MAC_TXOPTIONS_ACK into the *txOptions* filed. An eventual problem that has to be taken into account is the possibility of retransmitting the same packet infinite number of times. To solve this, the limit of retransmission is fixed to maximum 1 retransmission for every erased packet. If the relay doesn't respond with ACK for 2 times, the source node ignores the fact that the erased packet wasn't delivered and continues to send packets. In this case the source node transmit only information data.
In order to make fair comparision with the case in which FEC algorithm is used, the time used by the source node in the FEC to transmit an entire source block has to be equal to the time in which the source node transmit only information packets. For example, if the source node in the FEC employs 7 seconds to send 7 packets (information data and redundancy), the source node in the ARQ has to employ the same time to send only information data (1.35 seconds for each packet).
Although the **relay** is programmed to tolerate only one packet erasure and resend the not-acknowledged packet.

In the third case considered, all nodes are programmed to ignore the erasures introduced by the channel.

5.3.3 Results

The experimentation regards two different scenarios. In the first one, the nodes are placed in way that the main propagation path between them is not obstructed by any obstacle. Figure 5.6 illustrate the scenario. The blue rumble indicates the source node and the red one

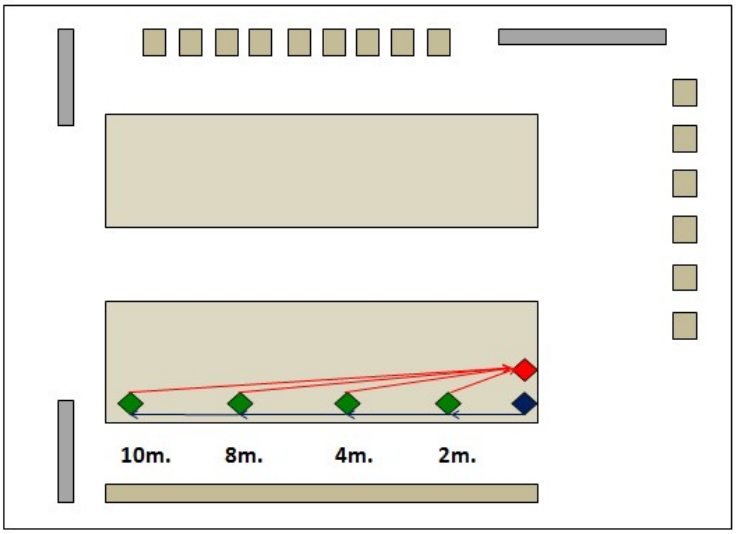

Figure 5.6: *The scenario without obstacles on the main propagation path*

the destination node. The relay (the green rumble) was placed on different positions, varying the distance from/to the source/destination node. The results of the experimentation are illustrated on Figure 5.7. From the graphic, it is possible observe that for this scenario the PLR grow as the distance increases. The red line present the worst case, a network at which all erased packets are ignored. With the introduction of the ARQ protocol, the PLR decreases significantly, as expected. Due to the limitation of number of retransmissions and the energy saving, also in this case the PLR is not completely reduced to zero packet losses. The green line presents the case at which the FEC algorithm is used. It shows further improvement of the PLR.

In the second scenario the relay is placed on three different positions and there are obstacles present on the main propagation path.

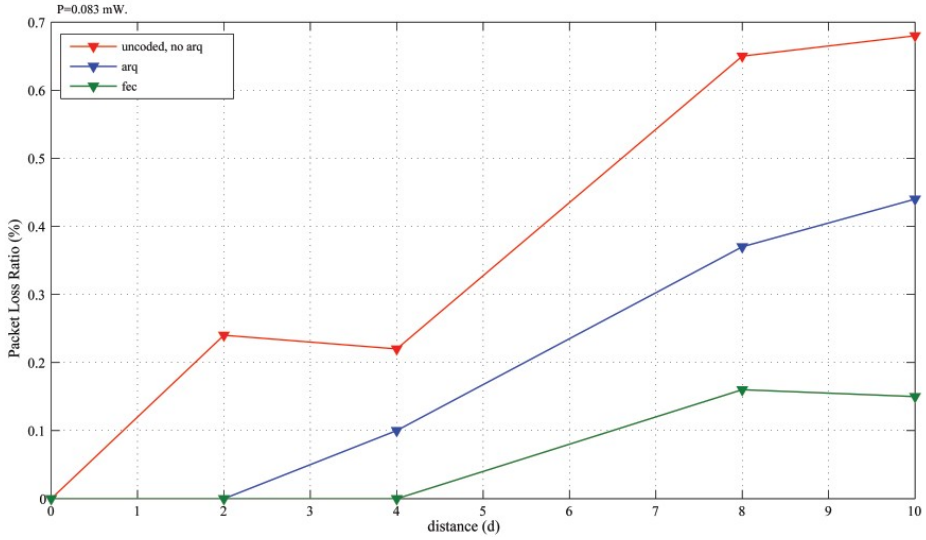

Figure 5.7: *The results in the first scenario without obstacles on the main propagation path*

This means that the measures in this scenario are more influenced by the propagation environment than the first scenario and the *Packet Loss Ratio* not necessary grows as the distance between the nodes increases. Figure 5.8 shows the scenario. The results obtained are presented on Figure 5.9.

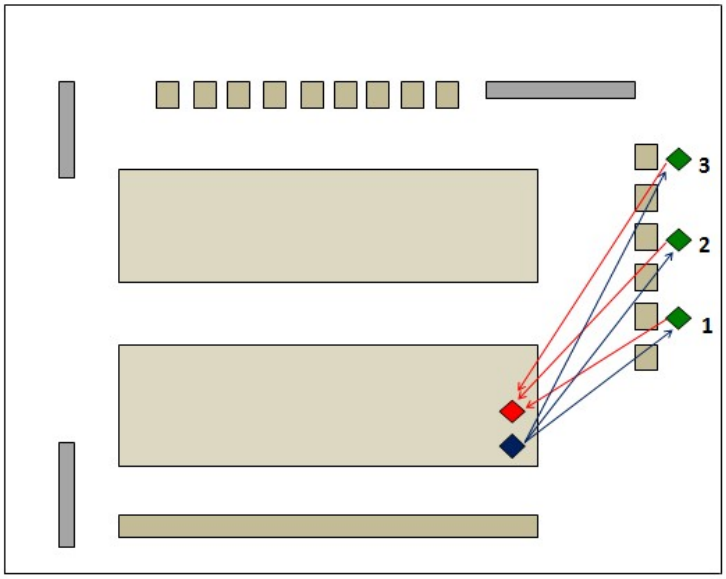

Figure 5.8: *The scenario with obstacles present on the main propagation path*

In comparison to the other scenario, in this case the PLR is increased due to the obstacles present on the main propagation path between the nodes and the FEC algorithm shows also the best performance. In both scenarios, the power transmission is the same (P=0.083 mW).

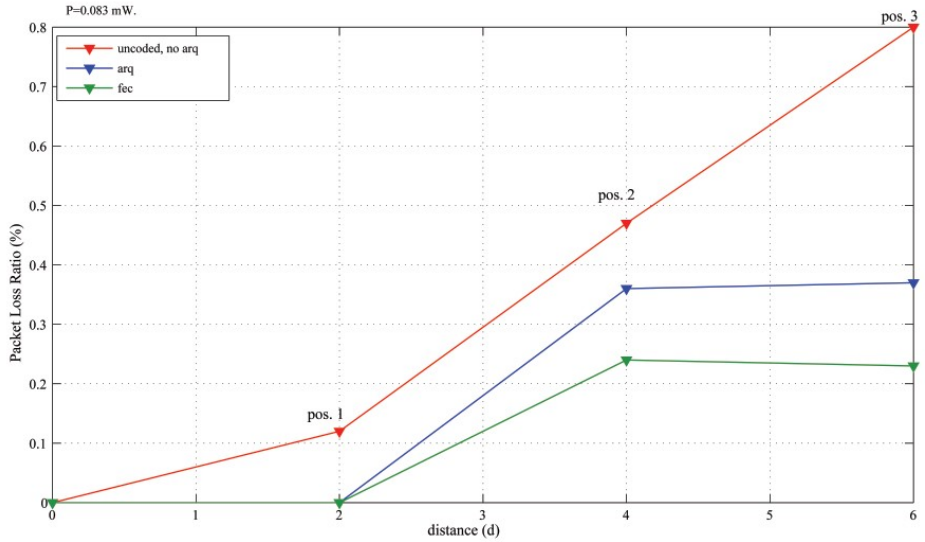

Figure 5.9: *The results in the second scenario with obstacles present on the main propagation path*

Before reaching the final conclusions, it is necessary make some quantitative comparisons between the FEC and the ARQ. To do that, in the ARQ case is necessary to know the total number of the retransmitted packets in order to evaluate how much extra energy was used for the retransmissions. On the other hand, in the FEC algorithm is important considerate that the real number of transmitted packets wasn't 100, but 175, including the redundancy packets. So, it must be taken into account the extra energy spend for the transmission of the redundancy packets. Unfortunately, the software tools available weren't enough for the registration of the retransmissions effected by the relay. So it wasn't possible to know the total number of retransmitted packets by this node. In this way, wasn't possible make some quantitative comparisons.

5.4 Conclusions

Before making a decision to implement a packet recovery scheme and prefer one more than another, it is necessary make evaluations between the extra energy spend for the transmission of the redundancy introduced and the capability of correction of the coding/decoding techniques and, on the other hand, the extra energy spend for the retransmissions and the limitation of the number of retransmissions allowed. In the scenarios considered, the graphics show that the implementation of the FEC algorithm allows significantly reduction of the PLR. The quantitative measures are those that allow to make the final conclusion and to choose between the ARQ protocol and the implemented FEC algorithm.

5.5 Problems observed

During the implementation of the program code for the realization of the coding/decoding techniques and the experimentation, it were detected several problems. For example, for the realization of the proposed multi-hop network in order to **know a priori the short address of the destination node**, it was necessary all nodes making part of the network to be associated to the PAN coordinator and not to other nodes. In other words, it was necessary disable the association permit parameter of each node that is not the PAN. Only in this way was possible obtain different short addresses for the nodes and to know a priori the short address assigned by the PAN coordinator. The PAN assign the short address during the association phase, according to the firing order of the nodes, using an array *Router_DevShortAddrList* containing the short addresses {0x0001, 0x0002, 0x0003....}. Thus, in order to avoid the association of a generic node to some other previously associated node that isn't the PAN, it was necessary set into the *Device_Startup()* function the ASSOCIATION_PERMIT attribute on false using *Mac_MlmeSetReq (ASSOCIATION_PERMIT,* Router_MACFalse*)*. In this way, into the *Router_McpsDataReq()* it was possible specify the short address of the receiver without doubts that it could be the wrong address.
Another problem observed is the **not correct and strange behav-**

ior of the nodes CC2430EM when connected to the battery support board (SOC BB) and not to the SMARTRF04EB (equipped with LCD). In the program code is enabled the writing on the LCD. Since the SOC BB isn't equipped with LCD, it is necessary disable some PREPROCESSOR SYMBOLS. To solve this, it is necessary modify the general options of the project. In the C/C++ Compiler, PREPROCESSOR, deleting the defined symbols *HAL_LCD =TRUE* and *LCD_HW=TRUE* enable the correct behavior of the node.

5.6 Future work

The applications implemented could be extended in a lot of different scenarios. For instance, the same experimentation could be made in a network composed by more than three sensor nodes or in a different topology of network, for instance the star-topology. Another interesting case to study is the evaluation of the average time of arrival of the packets for which the synchronization through the nodes is essential.

Bibliography

[1] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, and W. J. Kaiser. *Wireless Integrated Network Sensors: Low Power Systems on a chip.* In Proceedings of the 1998 European Solid State Circuits Conference, The Hague, Netherlands, 1998.

[2] www.wikipedia.it

[3] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. *Wireless Sensor Networks for Habitat Monitoring.* In Proceedings of the 1st ACM Workshop on Wireless Sensor Networks and Applications.Atlanta, GA, September 2002.

[4] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar.*Next Century Challenges: Scalable Coordination in Sensor Networks.*In Proceedings of the Fifth Annual International Conference on Mobile Computing and Networks (MobiCom 1999), Seattle, Washington, DC, 1999.

[5] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao.*Habitat Monitoring:Application Driver for Wireless Communications Technology.*In Proceedings of the ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean, San Jose, Costa Rica, 2001

[6] Holger Karl and Andreas Willig*Protocols and Architectures for Wireless Sensor Networks.*

[7] G. Asada, M. Dong, T. S. Lin, F. Newberg, G. Pottie, and W. J. Kaiser. *Wireless Integrated Network Sensors: Low Power Sys-*

tems on a chip. In Proceedings of the 1998 European Solid State Circuits Conference, The Hague, Netherlands, 1998.

[8] LAN/MAN Standards Committee of the IEEE Computer Society.*IEEE Standard for Information technology– Telecommunications and information exchange between systems Local and metropolitan area networks – Specific requirements* – Part 15.4: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (LR-WPANs)*, October 2003.

[9] P.G.M. Baltus and R. Dekker. *Optimizing RF Front Ends for Low Power. Proceedings of the IEEE*,88(10): 1546–1559, 2000.

[10] V.Raghunathan, C.Schurgers, S.Park, and M.B.Srivastava. *Energy-Aware Wireless Microsensor Networks. IEEE Signal Processing Magazine*,19: 40–50, 2002.

[11] A.Wang, S.-H.Cho, C.G.Sodini, and A.P. Chandrakasan. *Energy-Efficient Modulation and MAC for Asymmetric Microsensor Systems.*In Proceedings of ISLPED 2001, Huntington Beach, CA, August 2001.

[12] CC1000 Single Chip Very Low Power RF Transceiver.Chipcon Product Data Sheet. http://www.chipcon.com/files CC1000-Data-Sheet-2-1.pdf.

[13] CC2420 2.4 GHz IEEE 802.15.4 / Zigbee RF Transceiver. Chipcon Product Data Sheet. http://www.chipcon.com/files CC2420-Data-Sheet-1-0.pdf.

[14] Chipcon Products from Texas Instruments *CC2430DK Development Kit User Manual Rev.1.0*

[15] K.Chakrabarty, S.S.Iyengar, H.Qi, and E.Cho.*Coding Theory Framework for Target Location in Distributed Sensor Networks. In Proceedings of the International Symposium on Information Technology:Coding and Computing*, pages 130–134, Las Vegas, NV, 2001.

[16] R.Verdone, D.Dardari, G.Mazzani and Conti A.: *Wireless Sensor and Actuator networks, Techologies, Analysis and Design*

[17] J.Hill, R.Szewczyk, A.Woo, S.Hollar, D.E.Culler, and K.S.J.Pister.*System Architecture Directions for Networked Sensors.In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 93–104, Cambridge, MA, 2000.

[18] Di Jun Zheng (Ph. D.),Jun Zheng,Abbas Jamalipour: *Wireless sensor networks: a networking perspective.*

[19] E. Neiwisadomska-Szynkiewicz, P. Kwasniewski and I. Windyga: *Comparative Study of Wireless Sensor Networks Energy-Efficient Topologie and Power Save Protocols*, article, 2009.

[20] : Robert H.Morelos-Zaragoza *The Art of Error Correcting Coding* Sony Computer Science Laboratories, Inc.Jap.

[21] Chipcon Products from Texas Instruments:*802.15.4 MAC Application Programming Interface.*

[22] Chipcon Products from Texas Instruments:*HAL Drivers Application Programming Interface.*

[23] Chipcon Products form Texas Instruments:*Z-Stack OS Abstraction Layer Application Programming Interface.*

[24] User manual SmartRF cc2420DK:*Packet Sniffer for IEEE 802.15.4 and ZigBee*

[25] Sukun Kim: *Efficient Erasure Code for Wireless Sensor Networks*

[26] I. F. Akyildiz, W. Su, Y. Sankarasubrama, E. Cayciri, *"Wireless Sensor networks: A Survey,"* Computer Networks. Elsevier Journal, pp. 393-422, March 2002.

[27] R. Agarwal, E. M. Popovici and B. O'Flynn, *"Adaptive Wireless Sensor Networks: A System Design Perspective to Adaptive Reliability,"* In Wireless Communications and Sensor Networks, pp. 216-225, 2006.

[28] Heikki Karvonen, Zach Shelby and Carlos Pomalaza-R´aez, *"Coding for Energy Efficient Wireless Embedded Networks,"* In Proc. International Workshop on Wireless Ad-hoc Networks 2004 (IWWAN 2004), 31 May- 3 June 2004, 1-5 CD-Rom.

[29] Mihaela Van Der Schaar, Philip A.Chou: *Multimedia over IP and Wireless Networks, compression, networking and systems.*

List of Figures

1.1 *Overview of main sensor node hardware components* [6]. 7
1.2 *Energy consumption in a single node*[16] 14
1.3 *Two programming models for WSN operating systems:purely sequential execution (a) and process-based execution (b)*[6] 16
1.4 *Event-based programming model*[6] 17

2.1 *The protocol stack of the IEEE 802.15.4 and ZigBee Standards*[18]. 22
2.2 *WPAN device architecture*[8] 23
2.3 *Frequency bands and data rates*[8] 25
2.4 *Format of the PPDU[8]* 27
2.5 *Modulation and spreading functions[8]* 28
2.6 *The MAC sublayer reference model[8]* 29
2.7 *General MAC frame format*[8] 30
2.8 *Format of the Frame Control field*[8] 30
2.9 *Possible value of the Destination Addressing Mode and Source Addressing Mode subfields*[8] 32
2.10 *Communication to a coordinator in a beacon-enabled PAN[8]* 35
2.11 *Communication to a coordinator in a nonbeacon enabled PAN [8]* 36
2.12 *Communication from a coordinator in a beacon-enabled PAN[8]* 36
2.13 *Communication form a coordinator in a nonbeacon-enabled PAN[8]* 37
2.14 *Schematic view of the beacon frame and the PHY packet[8]* 38
2.15 *Schematic view of the data frame and the PHY packet[8]* 39

2.16 *Schematic view of the acknowledgment frame and the PHY packet[8]* 40
2.17 *Schematic view of the MAC command frame and the PHY packet[8]* 41
2.18 *Architecture of the ZigBee stack[16]* 42

3.1 *A canonical digital communications system[20]* 47
3.2 *Systematic block encoding for error correction[20]* . . . 47
3.3 *Mechanism of erasure code[25]* 48
3.4 *Encoding/decoding using Single Check Parity Code* . . 53
3.5 *Encoding and decoding using Hamming(7,4) Code* . . . 54
3.6 *A representation of the Binary Erasure Channel [29]* . 56
3.7 *Probability of error using Single Check Parity Code* . . 59
3.8 *Flow diagram of the PER simulation* 61
3.9 *Simulated evolution of PER using Single Check Parity Code* 62
3.10 *Theoretical evolution vs. simulated results in the estimation of PER* 63

4.1 *Hardware components*[14] 66
4.2 *CC2430EB evaluation board*[14] 66
4.3 *CC2430EM evaluation board*[14] 67
4.4 *Software environment IAR Embedded Workbench* . . . 70
4.5 *Non beacon-enabled network start*[21] 73
4.6 *Device connecting to a non beacon-enabled network*[21] 75
4.7 *Data transactions*[21] 76
4.8 *Packet Sniffer screenshot* 77
4.9 *The transmission chain using Single Check Parity Code* 80
4.10 *The packet generated by the Data_create() function* . . 80
4.11 *Encoding technique* 81
4.12 *MAC data frame containing the application data* 82
4.13 *MAC data frame arrived at the destination node* 82
4.14 *The decoding used in FEC Single Check Parity Code algorithm* 83
4.15 *Successful decoding in FEC Single Check Parity code algorithm* 85
4.16 *The transmission chain using Hamming (7,4) Code* . . 86

4.17 *The use of the arrays in the FEC Hamming code algorithm* 87
4.18 *Example of transmission with two packets erased* 89
4.19 *Example of transmission with two data packets erased* . 90
4.20 *Example of transmission using ARQ protocol* 91

5.1 *The network created* 94
5.2 *Simulated vs. Theoretical evolution of the Probability of Decoding Failure using Single Check Parity Code* 95
5.3 *Simulated vs. Theoretical evolution of the Probability of Decoding Failure using Hamming(7,4) Code* 96
5.4 *Confront of the simulated values and the theoretical evolution of the Probability of Decoding Failure* 97
5.5 *The network realized for the experimentation* 98
5.6 *The scenario without obstacles on the main propagation path* 100
5.7 *The results in the first scenario without obstacles on the main propagation path* 101
5.8 *The scenario with obstacles present on the main propagation path* 102
5.9 *The results in the second scenario with obstacles present on the main propagation path* 103

Acknowledgments

In first place, I would like to thank to my supervisor **Prof. Marco Chiani**, that has always demonstrated willingness, availability and interest for all the work done. Working under his supervision was an excellent experience that permits me discover new things, grow like a professional and student. Thank you Prof.

The second thanks goes to my second supervisors **Enrico Paolini**, and **Matteo Mazzotti** that have always demonstrated availability, professionalism and have followed continuously the presented work.

Thanks to **Francesco** and **Simone** for the help with the instruments used in the thesis and the discussions about the programming schemes.

This study experience was very significant to me. Useful, interesting, enthusiastic and difficult at the same time. It helps me grow up as a person, discover and learn new things, develop my technical and professional abilities, and the most important, has brought me many new persons in my life that I appreciate and want to mention here. In this way I can thanks them for every new thing they learn me during this trip. The thanks with the big "T" goes to my **gorgeous family**. Sorry, but this I want to write it on macedonian: Mamo, Tato i Bato. BLAGODARAM! Mi dadovte mnogu, najmnogu. Blagodaram za moznosta da bidam tuka, da steknam novi znaenja, sposobnosti, prijateli, iskustva. Zelbata (familjarno steknata) da se nadmine sekoja precka, napravi site momenti na nedostig da bidat samo mig na pominliva nostalgija. Vi blagodaram za vasata podrska, pomos, ljubov, pozrtvuvanost i iskrenost.

Another thanks goes to my **Mau**. Without him and his daily support and love I haven't been capable to arrive at this point. Thank you for your being there always, for your patience, trust, your comprehensibility and support. For all moments of happiness and love.

Thanks to my special and favorite informatics nerd **Andri**. Thank you for the way you are, for all your advices, the help and availability offered for the exams and the thesis, for the beautiful days spend together. Remember always: The resistors were parallel connected!

A special thanks goes to my favorite English professor and best friend **Marina**. Thanks for your patience, presence, for your support, positivism and availability to control my English.

Thanks to **Anna**. My favorite university mate. Thank you for our collaboration during this university experience, for the lent notes, for the discussions made, for the preparation of the exams, for the jokes during the laboratory lessons, for the coffee breaks and the countless lunches spend together.

The last, but not less important thanks goes to all my special friends that I love and respect. Without them, these years passed in Italy weren't the same thing. Thanks for all moments passed together. Thanks to: **Gianluca P.S.**, , **Giuseppe**, **Silvia**, **Marry**, **Cecilia**, **Giusi**, **Fiona**,**Tanja**, **Ika**, **Simon**, **Cristian**, **Francesco**, **Simone R.**, **Vincenzo**, **Silvia M.**, **Alice**, **Laura**, **Anna Z.**, **Gloria**, **Nadia** and all others that i haven't mentioned.