

UNIVERSIDAD DE SEVILLA

ESCUELA SUPERIOR DE INGENIEROS

Proyecto Final de Carrera
Ingeniería de Telecomunicación

ADAPTIVE SOURCE AND CHANNEL CODING ALGORITHMS FOR ENERGY SAVING IN WIRELESS SENSOR NETWORKS

Tutor:

Dr. Ing. F.J. Payán Somet

Candidato:

Benigno Zurita Ares

I hereby encourage the reader to forgive and indicate any possible mistake, omission or inaccuracy that could find along this work.

Author
Benigno Zurita Ares

Universidad de Sevilla
Escuela Superior de Ingenieros

Dept. de Teoría de la Señal y Comunicaciones
Camino de los Descubrimientos, s/n
41092 Sevilla, España

Email: benigno.zurita@gmail.com
Web: www.us.es, www.esi.us.es

To my parents

Abstract

One of the major challenges to design efficient Wireless Sensors Networks (WSN) is the scarcity of energy and computational resources. We address this problem with particular reference to algorithms for efficient source and channel coding.

Distributed source coding (DSC) is a general framework which applies to highly correlated signals that are coded separately and decoded jointly. In WSN, DSC schemes provide closed loop algorithms that exploit the correlation of data sensed by the nodes to reduce the amount of information that each node transmits, thus saving energy. A study is herein carried out along with an implementation of the aforementioned algorithms with the Matlab environment. The stability of the closed loop algorithms is then tested by means of simulations.

Minimum energy coding schemes can be superimposed on top of the previous DSC algorithm to achieve further gains. In our work, we investigate and extend two existent energy-efficient minimum energy coding schemes [1][2]. In this context, we analyze the performance of the WSN in terms of power consumption and bit error probability, where a detailed wireless channel description is taken into account. We characterize the problem of efficient coding by means of stochastic optimization problems. This more accurate model of the system (compared to those previously existent [1][2]) allows us to propose new solutions to reduce power consumption while ensuring adequate bit error probabilities.

As a relevant part of our work, a test-bed has been set up by using the Berkeley Telos Motes, along with a Matlab application interface, for the adaptive source coding algorithm. According to the results obtained by the

experimental work we have carried out, the energy consumption can be effectively reduced.

Contents

List of Figures

List of Tables

Acknowledgements

Once more, I find myself in front of the defiant screen of my computer, who looks at me wondering what kind of discussion I will start typing on him today. Don't worry my dear friend, I guess today is a different day, different from all the days of the past five years; is the day in which my studies are concluded, the day in which I finally defend my Master Thesis.

And here I am, using the only section where the words "tradeoff" and "performance" are completely unnecessary. The only section where engineers are allowed to write about their feelings and their life.

It all started five years ago, when I faced the question, what should I study now? The answer was straightforward: you all know it. Durante estos cinco años trabajé muy duro, todos lo sabéis. Me dediqué en cuerpo y alma a obtener aquello que tanto anhelaba, y ahora estáis conmigo en el último paso, que no es último sino primero.

En la Escuela me formé, but it was not Fourier who made me wake up full of illusion every morning. Fuistéis vosotros: Manu (the boss...siempre estuviste ahí), Bruno (tú sólo estabas de vez en cuando), Fani (la alegría del grupo), Natalia (la cordobesa que más quiero), Inma (por la paz de tu ser), Óscar (por las Diosas), Sol (que siempres alumbres a mi mejor amigo) y Pepe (por nuestras discusiones en las que creíamos que podríamos cambiar el mundo). Vosotros...y muchos otros que no caben en el papel.

Countless all the things we did together: las clases de Celestino y su bata en primero (por aquel entonces Manu era rubio), en segundo las de Chávez (nunca imaginé que llegaría a reirme de chistes donde apareciese la palabra electrón) y las de Justo (y aquél día que llegaste tarde, Fani...), el año de tercero con "Los Serrano" y la "mirada del tigre", las mañanas de cuarto

con todas las leyes de Murphy habidas y por haber en Tratamiento Digital de Señales, aquél Matalascañas en el que descubrimos que los bocadillos de tortilla y arena no estaban tan malos, nuestros paseos por el río. Las noches interminables por Nervión, Plaza Cuba, La Alfalfa y la Calle Betis. Las cervezas y las risas.

Thanks to you all, I really miss you.

Time went fast, and the last year of my studies arrived, I decided to "fly out of the nest". I left my family, my friends and my comfortable and warm Sevilla behind me, defying life and weather: I came to Sweden.

It has been an intense year where I had the luck of meeting incredible people, find new friends and even love. If I tried to write here the names of all of you, I would never finish. So I decided to just write three names (hard choice here). The first one is you, Isa, because you are my one. Pablo, you come second, I could say as many things of you as spaghetti we have eaten together. I will limit myself to just one word: forever...and you know you will end up in Spain, don't you? Fran, it's a pity you left Sweden so soon, I always expect to see you again, entering the kitchen with a couple of beers. For the rest: Rosa, Xavi, Gabri, Elenas (en plural), Jordi, Ana, Marta, Joan...you know how important you are for me. Thanks for insistingly phoning me even when I disappeared during days, and even weeks (at the end, they have been more than three names...).

I would like to thank my Master Thesis advisors, Prof. Karl Henrik Johansson and Dr. Carlo Fischione. I'm very grateful for the opportunity I had to work with you at the Kungliga Tekniska Högskolan (KTH), Stockholm, Sweden. Thank you for all the things you taught me, and for your continuous support and encouragement. Thanks also for the position you have offered me, I'm not going to disappoint you.

I would also like to express my gratitude to Dr. F. Javier Payán from Universidad de Sevilla, Spain, for his endless encouragement and for guiding me in the complex world of the data transmission theory. Without his teachings I wouldn't have been able to carry out this work.

To finish with, I would like to thank my family, because they are so few but so much: papá y mamá, tía Isabel, Camino, Inés, Jesús, Carlos y Juanito.

No imagino una vida sin vosotros. Gracias por el apoyo que me habéis dado durante estos cinco años, por los mareos a los santos, por las tardes de aburrimiento común en Sevilla, por las largas horas de teléfono, por vuestra alegría, vuestro cariño y vuestro amor...por ser vosotros: gracias por darme la vida.

Introduction

Recent advancement in the semiconductor industry has enabled the development of small, inexpensive, low-powered devices known as sensor nodes. These sensors are endowed with data sensing, data processing and communication components that convert them in multi-functional, flexible platforms.

A wireless sensor network (WSN) is composed of many of these tiny sensor nodes that collaborate to accomplish a common task and are densely deployed in the area to be monitored. A broad variety of applications ranging from geophysical monitoring (seismic activity) to precision agriculture (soil management), habitat and environmental monitoring, military systems and business processes (supply chain management) are supposed to be implemented on WSN.

Unlike traditional wireless networks and ad hoc networks, WSN feature dense node deployment, unreliable sensor nodes, frequent topology change, and severe power, computation and memory constraints. These unique characteristics pose many new challenges to practical realization of WSNs, such as energy conservation, self-organization, fault tolerance, etc. In particular, sensor nodes are usually battery-powered and should operate without attendance for a relatively long period of time. In most scenarios, it is very difficult and even impossible to change or recharge batteries. For this reason, energy efficiency is of primary importance for the operational lifetime of a sensor network.

The wireless medium is used for communication in WSN. However, the wireless nature of the channel forces to deal with undesired phenomena as path losses, channel fading, interferences, and noise disturbances, which

cause packets losses, transmission errors and serious delays in the data reception. Therefore, the effect of the wireless channel must be considered when designing energy efficient WSN.

Previous considerations showed the necessity of suitably addressing the energy consumption problem. Many efforts have been made in this direction, and many researchers in the scientific community are currently involved in some specific aspects as:

- Energy-efficient network protocols: most existing network protocols and algorithms for traditional wireless ad hoc networks cannot effectively address the power constraint and other constraints of sensor networks. To realize the vision of sensor networks, it is imperative to develop various energy-efficient network protocols in order to efficiently use the limited power in each sensor node and prolong the lifetime of the network.

- Power control: a suitable tuning of the nodes' transmission power helps to reduce the overall network consumption by transmitting at lower power levels when possible.

- Topology control: this technique consists of letting a subset of the nodes to sleep, while others are active. Topology control reduces the redundancy present in the network as well as the interferences, as lesser number of nodes are active in any given neighborhood, which helps to reduce the Multiple Access Interference (MAI) and consequently, the necessary transmission power.

- Distributed and adaptive source coding: as a result of the high deployment density, nodes sense highly correlated data containing both spatial and temporal redundancy. Thus, given the high correlation present in the data, is important to implement suitable protocols and source coding techniques that exploit this particular characteristic to reduce the overall transmitted information (i.e. to lower the energy consumption). Nevertheless, algorithms must remain simple enough to be able to fit the scarce memory and the low capacity of the sensors' processing unit.

- Energy-efficient source+channel coding: as previously remarked, the dense deployment of the sensors (few meters is the typical distance between nodes) causes serious problems in the communication since nearby nodes can overwhelm the received signal of the desired sensor node forcing to increase the transmission power. CDMA is a promising multiple access scheme for sensor and ad hoc networks due to its interference averaging properties [10]. However, the performance of CDMA systems is limited by MAI. In the past decade numerous methods have been developed to reduce MAI, most of which focus on the design of effective correlation receivers. However, they also introduce an increase in complexity, which is an undesired effect for WSN. Instead of merely designing receivers to suppress interferences, these techniques try to smartly represent the output of the source with a special codebook so that MAI is greatly reduced.

In our work we address the last two issues of the previous list. In Chapter 1 we study a distributed and adaptive source coding algorithm with which we exploit the existing redundancy in the sensed data in the network to locally process the measured data so that we reduce the actual information that needs to be sent over the wireless channel (and so, the power consumed); in Chapter 2 we introduce a source+channel coding scheme, and we state and solve a constrained power minimization algorithm. We also carry out a theoretical study of the system performance in terms of power consumption and bit error probability, considering the presence of the wireless channel; in Chapter 3 we step forward and we present a more complex source+channel coding scheme; novel expressions for the power consumption and bit error probability are derived; in Chapter 4 we report the results of implementing the distributed source coding algorithm studied in Chapter 1 in a real WSN; Finally, in Chapter 5, we conclude our work with the presentation of the conclusions and the outline for the future work.

Chapter 1

Distributed and Adaptive Source Coding

1.1 Introduction

Advances on the electronics and wireless networking have enabled the appearance of new tiny devices endowed with many different types of sensors and capabilities. They are going to make possible the beginning of new exciting applications that will open doors previously closed to the human being. This nodes are usually small in physical dimensions and operated by battery power. Since in most cases access to the sensors once they have been deployed is extremely hard or in many cases almost impossible, it is easy to understand that any technology that make possible energy savings is welcome. Thus, a big effort has been made by the research and scientific community in order to reduce energy consumption in such networks. In this chapter, a method proposed by Petrovic, Ramchandran and Chou [3] is studied.

An implementation on a real Wireless Sensor Network (WSN) was also carried out. Results are presented in Chapter 4.

The studied algorithm is based on the adaptive signal processing theory as well as on distributed source coding. The main underlying idea consists of taking advantage of the correlation brought about by the spatio-temporal characteristics of the physical medium to reduce the amount of

information requested to the sensors. Furthermore, this scheme is orthogonally different from other studies carried out in the area, allowing a total integration with other energy-aware technologies like packet/data aggregation (reduces the overhead in the network) [4][5][6], efficient information processing [7][8].

Two factors of correlation should be highlighted and, consequently, considered for the correlation tracking algorithm:

- Measures taken by sensors closely located show a similar pattern.
- The signals being sensed (temperature, humidity,...) follow some basic rules of continuity and/or statistical properties.

Dense networks offer data samples highly correlated, because the sensors used are physically placed closely, with the consequent redundance obtained in the measures. An example of this can be a WSN deployed to measure the temperature and humidity in the ecosystem created around a tree, with the sensors measuring the evolution of these parameters in different parts of the tree. Another example can be the recording of audio data [3] as the cry of the whales or even a concert. Audio is a very suitable data source to apply this algorithm, due to the intrinsic presence of redundance (echoes).

Petrovich's algorithm is applied in a network consisting on two types of nodes: many sensing nodes and one or more data-gathering nodes (see Fig. 1.1). The former ones are supposed to be energy constrained, not so the latter. One of the most appealing characteristics of this algorithm is that no data exchange between sensors is needed, the compression takes place in a fully blind manner, with the subsequent saving of energy. These savings are achieved by letting the data gathering node track the correlation structure among nodes and process the information to effect distributed sensor data compression. The correlation structure is determined by using an adaptive prediction algorithm. The sensors, in turn, only need to know the number of bits they are entitled to send to the data-gathering node.

It is obvious the complexity in the decoder is higher than that in the encoder, but it resides on the data-gathering node, which is assumed not to

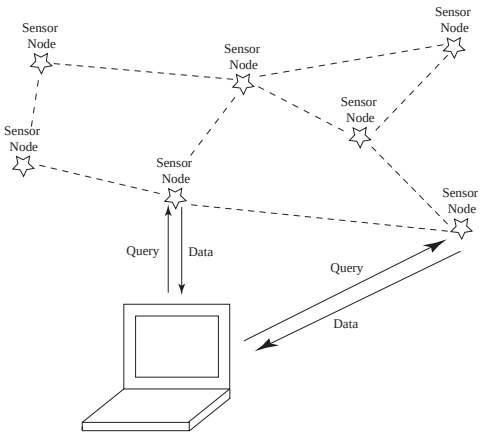

Figure 1.1: *An example of a WSN in which the laptop acts as the sink node gathering the information requested to the sensors.*

be energy constrained.

1.2 Distributed compression

Let us focus now in how distributed compression is achieved. Say we have $n + 1$ nodes, where one of them behaves as a data-gathering node. The algorithm works as follows: when the sink node starts to request information it has no information at all which allows it to perform redundance elimination, thus, it begins asking for uncompressed data. Once it is in possession of data coming from the sensors it is able to reduce the amount of information requested to the sensors, since this information will be partially redundant with the one previously received. The data-gathering node uses this old information to obtain a prediction of the next value it is going to request.

Let us imagine it is turn to query sensor j and we are at time instant k. The desired data can be represented by $X_k^{(j)}$. The corresponding estimate is denoted by $Y_k^{(j)}$ and is calculated by means of the correlation tracking algorithm. This algorithm uses previous measures of sensor j as well as measures belonging to neighboring sensor nodes. All measures are conveniently weighted to yield the optimal estimate in terms of a particular criterion (in this case we chose to minimize the mean square error). Based

on the accuracy of this prediction and some other parameters as the desired probability of error, the sink node will require a specific number of bits (say m) to sensor j which, in turn, will transmit the data sensed (converted by the ADC to n bits) compressed to those m bits. The decoder will have to decode this compressed sequence of bits to the desired data message, $X_k^{(j)}$, for what it uses the value $Y_k^{(j)}$ previously calculated.

The correlation tracking algorithm must address an important issue: data statistics might be time-varying and consequently, the amount of correlation can change. This has two consequences, first, the correlation tracking algorithm can not be static and the coefficients weighting the different data have to be updated with the sufficient frequency. Second, it is mandatory to have one unique underlying codebook that it is not changed among the sensors and can also support multiple compression rates. This concept can be pictured as having several codebooks forming a tree structure as shown in Fig. 1.2.

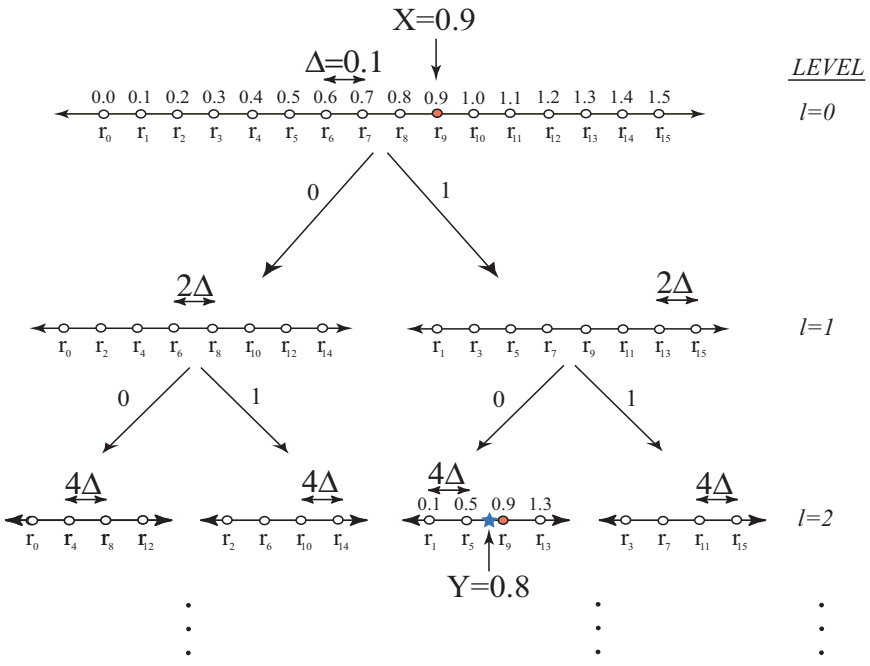

Figure 1.2: *An example for the tree based codebook. The encoder is asked to encode X using 2 bits, so it transmits $\{01\}$ accordingly to the decoder. The decoder will use the bits $\{01\}$ in an ascending order from the least-significative-bit (LSB) to determine the path to the subcodebook to use to decode Y.*

During the remaining of the chapter we will repeatedly talk about codewords and codebooks, so the reader should have a clear understanding of what is meant by these two terms. By codeword we denote the binary representation for each of the symbols coming out from the quantizer. The codebook is just the set of all possible codewords. Using the same nomenclature as in Appendix A, we can say that the codebook is the binary representation of the alphabet whereas the codewords are the binary representations of the letters. Some times, we talk about codebooks at a particular level, this is an abuse of language by which we denote the set of possible codewords in that determinate level. For example, in Figure 1.2, a codeword is represented by r_j ($j \in \{0, \cdots, 15\}$) while the codebook at level 0 is the set $\{r_0, r_1, r_2, r_3, \cdots, r_{14}, r_{15}\}$. In Figure 1.2, the distance between codewords at level i is $2^i\Delta$.

Let us briefly illustrate, now, how encoder and decoder work:

- Encoder: The encoding operation is carried out by the sensing nodes. It is preceded by the computation (in the sink node) of the number of necessary bits i that are going to be requested to the sensors for the data to be compressed. Once this number is computed (based on different parameters as probability of error, accuracy of previous predictions,...) the sensor is asked to send its information compressed down to this number of bits. Suppose the ADC returns the data sensed, $X_k^{(j)}$, with n bits. The mapping from $X_k^{(j)}$ to the compressed message with the specified number of bits can be done through the following deterministic mapping (1.1), resulting in new codewords belonging to a subcodebook at level i.

$$f(X_k^{(j)}) = \text{index}(X_k^{(j)}) mod 2^i \tag{1.1}$$

  What we basically do with this mapping is to keep the i least significative bits of the original data $X_k^{(j)}$. That is, we assume that the estimated value of $X_k^{(j)}$ is sufficiently accurate to predict the overall behavior of the sensed data so that it is only needed a small amount of information from the nodes to precisely determinate the value $X_k^{(j)}$. This way of proceeding is called decoding with side-information, where the side-information is the predicted value of $X_k^{(j)}$, $Y_k^{(j)}$, computed at

the decoder (the interested reader can find more information on this topic in Appendix A).

- Decoder: The decoding operation is, obviously, performed at the data-gathering node. When this node receives the requested data it starts to traverse the tree using the information received to locate the appropriate subcodebook $\mathcal{S}$ among all the possible ones at the level i at which the data has been compressed to. This process starts with the least-significant-bit (LSB) of $f(X_k^{(j)})$. Once the suitable subcodebook has been found, the decoder will use the side-information, $Y_k^{(j)}$, to decode the closest value in $\mathcal{S}$:

$$f(X_k^{(j)}) = \text{argmin}_{r_i \in \mathcal{S}} \| Y_k^{(j)} - r_i \| \tag{1.2}$$

  where r_i stands for the i^{th} codeword in $\mathcal{S}$.

If we study more carefully how the tree structure of subcodebooks is constructed, we can appreciate that at level i, the different codewords within the same subcodebook share the last i bits. This can be directly interpreted as the fact that the estimate $Y_k^{(j)}$ is not distinguishable among words of code belonging to level $i+1$ (or higher) so that it is necessary to have a finer decomposition. Thus, buy asking for the i least significative bits, we are able to rule out some codewords so that our subcodebook is small enough to guarantee a correct decoding from the side-information.

In short, we assure that the prediction $Y_k^{(j)}$ differs in less than $2^{i-1}\Delta$ from $X_k^{(j)}$ and we choose a subcodebook $\mathcal{S}$ whose codewords are at a distance $2^i\Delta$ from each other. In this way, a unique and successful decoding is achieved.

Let us illustrate the coding/decoding process with a simple example. Assume we have the tree structure given by Fig. 1.2, where each codeword in the codebook at level $l = 0$ is composed by four bits (the codebook has sixteen elements). Let us also assume that the data-gathering node has in some way computed the side-information $Y_k^{(j)}$ and the number of bits to which the data has to be compressed is $i = 2$. The process starts when the sink node requests data to sensor j. The sensor node uses its ADC to

obtain a 4-bit value $X_k^{(j)} = 0.9$ corresponding to codeword r_9. The following thing the sensor node does is to use the rule given by (1.1) to find the mapping in the compressed domain: $f(X_k^{(j)}) = 9 \text{mod} 4 = 1$. Thus, the encoder will send the two bits, $\{01\}$, to the k-th sink node. The sink node will make use of the received message to descend by the tree and find the suitable subcodebook $\mathcal{S}$ where to apply (1.2). It starts using the LSB '1' so that it breaks the root codebook (i.e., the codebook at level 0) down to $\{r_1, r_3, r_5, r_7, r_9, r_{11}, r_{13}, r_{15}\}$. Afterwards it uses the second bit in the message '0' to choose codebook $\{r_1, r_5, r_9, r_{13}\}$ as $\mathcal{S}$. It is now when the decoder uses the side-information $Y_k^{(j)}$ conveniently introduced in Eq. (1.2)

$$f(X_k^{(j)}) = \text{argmin}_{r_i \in \mathcal{S}} \{0.7, 0.3, 0.1, 0.5\}$$

to derive r_9 as the decoded codeword, which is exactly the value sensed in the node. Thus, we can see how transmission of the information has been achieved by using only 2 bits instead of 4, as it would have been needed without encoding.

1.3 Correlation tracking

In the previous section we assumed that some information, $Y_k^{(j)}$, correlated to the sensors readings, $X_k^{(j)}$, was available at the decoder for sensor j at time k. The prediction can be done by a linear combination of different measures available at the decoder and can be expressed as Eq. (1.3):

$$Y_k^{(j)} = \sum_{l=1}^{M} \alpha_l X_{k-l}^{(j)} + \sum_{i=1}^{j-1} \beta_i X_k^{(i)} \tag{1.3}$$

We can think of $Y_k^{(j)}$ as a linear prediction based on past values of the sensor whose measure is going to be predicted along with current values of neighboring sensors. Hence, the main objective of the decoder is to derive a good estimate of $X_k^{(j)}$ for each sensor j.

To find the values α_l and β_i which minimize the mean square error (MSE), a mathematical problem must be addressed. Let us start by representing the prediction error as a random variable, $N_j = Y_k^{(j)} - X_k^{(j)}$. We can expand

the mean square error as:

$$\begin{aligned} E\left[N_j^2\right] &= E\left[\left(X_k^{(j)} - \left(\sum_{l=1}^{M}\alpha_l X_{k-l}^{(j)} + \sum_{i=1}^{j-1}\beta_i X_k^{(i)}\right)\right)^2\right] \\ &= E\left[X_k^{(j)^2}\right] - 2\sum_{l=1}^{M}\alpha_l E\left[X_k^{(j)}X_{k-l}^{(j)}\right] \\ &\quad -2\sum_{i=1}^{N}\beta_i E\left[X_k^{(j)}X_k^{(i)}\right] + 2\sum_{l=1}^{M}\sum_{i=1}^{j-1}\alpha_l\beta_i E\left[X_{k-l}^{(j)}X_k^{(i)}\right] \\ &\quad + \sum_{l,h=1}^{M}\alpha_l\alpha_h E\left[X_{k-l}^{(j)}X_{k-h}^{(j)}\right] + \sum_{i,h=1}^{j-1}\beta_i\beta_h E\left[X_k^{(i)}X_k^{(h)}\right] \quad (1.4) \end{aligned}$$

Now, if we assume that $X_k^{(j)}$ and $X_k^{(i)}$ are pairwise jointly wide sense stationary for $i = 1, \ldots, j-1$ then we can rewrite the mean square error as:

$$E[N_j^2] = r_{x^j x^j}(0) - 2\mathbf{P}_j^T\mathbf{\Gamma}_j + \mathbf{\Gamma}_j^T R_{zz}^j \mathbf{\Gamma}_j \quad (1.5)$$

where the superscript T stands for the transpose and

$$\mathbf{\Gamma}_j = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_M & \beta_1 & \beta_2 & \cdots & \beta_{j-1} \end{bmatrix}^T,$$

$$\mathbf{P}_j = \begin{bmatrix} r_{x^j x^j}(1) & r_{x^j x^j}(2) & \cdots & r_{x^j x^j}(M) & r_{x^j x^1}(0) & r_{x^j x^2}(0) & \cdots & r_{x^j x^{j-1}}(0) \end{bmatrix}^T$$

and we use the notation $r_{x^j x^i}(l) = E[X_k^j X_{k+l}^i]$. We can express R_{zz}^j as:

$$R_{zz}^j = \begin{bmatrix} R_{x^j x^j} & R_{x^j x^i} \\ R_{x^j x^i}^T & R_{x^i x^i} \end{bmatrix} \quad (1.6)$$

where, in turn, $R_{x^j x^j}$ is given as:

$$R_{x^j x^j} = \begin{bmatrix} r_{x^j x^j}(0) & r_{x^j x^j}(1) & \cdots & r_{x^j x^j}(M-1) \\ r_{x^j x^j}(1) & r_{x^j x^j}(0) & \cdots & r_{x^j x^j}(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{x^j x^j}(M-1) & r_{x^j x^j}(M-2) & \cdots & r_{x^j x^j}(0) \end{bmatrix} \quad (1.7)$$

and $R_{x^j x^i}$ and $R_{x^i x^i}$ are given as:

$$R_{x^j x^j} = \begin{bmatrix} r_{x^j x^1}(1) & r_{x^j x^2}(1) & \cdots & r_{x^j x^{j-1}}(1) \\ r_{x^j x^1}(2) & r_{x^j x^2}(2) & \cdots & r_{x^j x^{j-1}}(2) \\ \vdots & \vdots & \ddots & \vdots \\ r_{x^j x^1}(M) & r_{x^j x^2}(M) & \cdots & r_{x^j x^{j-1}}(M) \end{bmatrix} \tag{1.8}$$

$$R_{x^i x^i} = \begin{bmatrix} r_{x^1 x^1}(0) & r_{x^1 x^2}(0) & \cdots & r_{x^1 x^{j-1}}(0) \\ r_{x^2 x^1}(0) & r_{x^2 x^2}(0) & \cdots & r_{x^2 x^{j-1}}(0) \\ \vdots & \vdots & \ddots & \vdots \\ r_{x^{j-1} x^1}(0) & r_{x^{j-1} x^2}(0) & \cdots & r_{x^{j-1} x^{j-1}}(0) \end{bmatrix} \tag{1.9}$$

To find the optimal set of coefficients (represented by $\mathbf{\Gamma}_j$) that minimizes the mean square error, we differentiate Eq. (1.5) with respect to $\mathbf{\Gamma}_j$ to obtain:

$$\begin{aligned} \frac{\partial E[N_j^2]}{\partial \mathbf{\Gamma}_j} &= \frac{\partial \left(r_{x^j x^j}(0) - 2\mathbf{P}_j^T \mathbf{\Gamma}_j + \mathbf{\Gamma}_j^T R_{zz}^j \mathbf{\Gamma}_j \right)}{\partial \mathbf{\Gamma}_j} \\ &= \frac{\partial \left(-2\mathbf{P}_j^T \mathbf{\Gamma}_j \right)}{\partial \mathbf{\Gamma}_j} + \frac{\partial \left(\mathbf{\Gamma}_j^T R_{zz}^j \mathbf{\Gamma}_j \right)}{\partial \mathbf{\Gamma}_j} \\ &= -2\mathbf{P}_j + \left[R_{zz}^{jT} + R_{zz}^j \right] \mathbf{\Gamma}_j \\ &= -2\mathbf{P}_j + 2R_{zz}^j \mathbf{\Gamma}_j \end{aligned} \tag{1.10}$$

Where basic matrix calculus has been employed:

$$\frac{\partial (Ax+b)^T C\,(Dx+e)}{\partial x} = A^T C\,(Dx+e) + D^T C^T\,(Ax+b) \tag{1.11}$$

Please note that in the formula above (1.11) upper case letters denote matrixes while vectors are written in the lower case.

Setting (1.10) equals zero and solving, we obtain the standard Wiener estimate:

$$\mathbf{\Gamma}_{j,opt} = R_{zz}^{-1,j} \mathbf{P}_j \tag{1.12}$$

If the assumption of stationarity held, then the data gathering could request for uncoded data from all the sensors for the first K rounds of requests and construct the correlation matrixes for their subsequent use in (1.12). By doing this we would already have tracked the behavior of the system and we

could employ the obtained set of coefficients for computing the side information for each future round of requests. Thus, it would be only necessary to calculate the standard Wiener estimate once, what would be really convenient given the extreme computational complexity of this calculus.

In practice, however, the statistics of the data may be (and actually they are) time varying and as a result, the coefficient vector, $\mathbf{\Gamma}_j$, must be continuously adjusted to minimize the mean square error. One method of doing this is to move $\mathbf{\Gamma}_j$ in the opposite direction of the gradient of the objective function (i.e., the mean squared error) for each new sample received during round $k+1$ (this method known in the literature as the 'Steepest Descent Method'):

$$\mathbf{\Gamma}_j^{(k+1)} = \mathbf{\Gamma}_j^{(k)} - \mu \nabla_j^{(k)} \tag{1.13}$$

where $\nabla_j^{(k)}$ is given by Eq. (1.10) and μ represents the step size of the Steepest Descent Method. The goal of this approach is to descend to the global minima of the objective function. We are assured that such a minima exists because the objective function is convex. In fact, it has been shown that if μ is chosen correctly then (1.13) will converge to the optimal solution.

From (1.10) and (1.13) can be shown that the coefficient vector should be updated following the rule:

$$\mathbf{\Gamma_j^{(k+1)}} = \mathbf{\Gamma}_j^{(k)} - \frac{1}{2}\mu \left(-2\mathbf{P}_j + 2R_{zz}^j \mathbf{\Gamma}_j^{(k)}\right) \tag{1.14}$$

However, in practice, the data gathering node will not have knowledge of $\mathbf{P}_j$ and R_{zz}^j due to the computational complexity required for obtaining them. Hence, it will be necessary to provide the algorithm with an efficient method for estimating $\mathbf{P}_j$ and R_{zz}^j. One standard estimate is to use $\mathbf{P}_j =$

$X_k^{(j)}\mathbf{Z}_{k,j}$ and $R_{zz} = \mathbf{Z}_{k,j}\mathbf{Z}_{k,j}$ with

$$\mathbf{Z}_{k,j} = \begin{bmatrix} X_{k-1}^{(j)} \\ X_{k-2}^{(j)} \\ \cdots \\ X_{k-M}^{(j)} \\ X_k^{(1)} \\ X_k^{(2)} \\ \cdots \\ X_k^{(j-1)} \end{bmatrix}$$

introducing these new terms, Eq. (1.14) remains as

$$\begin{aligned} \mathbf{\Gamma}_j^{(k+1)} &= \mathbf{\Gamma}_j^{(k)} - \mu\mathbf{Z}_{k,j}(-X_k^{(j)} + \mathbf{Z}_{k,j}^T\mathbf{\Gamma}_j^{(k)}) \\ &= \mathbf{\Gamma}_j^{(k)} + \mu\mathbf{Z}_{k,j}N_{k,j} \end{aligned} \tag{1.15}$$

where the second equality follows from the fact that $Y_k^{(j)} = \mathbf{Z}_{k,j}^T\mathbf{\Gamma}_j^{(k)}$ and $N_{k,j} = X_k^{(j)} - Y_k^{(j)}$.

In practice the formulas above yield the following practical equations (well known in the adaptive filtering literature as the Least-Mean-Squares (LMS) algorithm):

1. $Y_k^{(j)} = \mathbf{\Gamma}_j^{(k)T}\mathbf{Z}_{k,j}$
2. $N_{k,j} = X_k^{(j)} - Y_k^{(j)}$
3. $\mathbf{\Gamma}_j^{(k+1)} = \mathbf{\Gamma}_j^{(k)} + \mu\mathbf{Z}_{k,j}N_{k,j}$

To use this algorithm, the data-gathering node will start asking the sensing nodes for sending their data uncompressed during the first K rounds. By doing this we ensure that the algorithm converges. Once this phase has been carried out, we can consider that we have tracked the correlation structure existing between the different nodes, so we can start to ask for compressed data. This is done in two steps (as already shown): first, computing the prediction of the data to be requested, for what we know that $Y_k^{(j)} = \mathbf{\Gamma}_j^{(k)T}\mathbf{Z}_{k,j}$ (recall that the statistics of the sources can change with the time, so the vector of coefficients has to be continuously updated). The

second step mentioned before, consists of obtaining the number of bits for the data to be compressed to.

The decoder, in turn, will decode $Y_k^{(j)}$ to the closest codeword in the sub-codebook $\mathcal{S}$ as has already been seen, yielding $\hat{X}_k^{(j)}$. From section 1.2 we know that while $2^{i-1}\Delta > |N_{k,j}|$ and the encoder encodes $X_k^{(j)}$ with i bits, then, $\hat{X}_k^{(j)} = X_k^{(j)}$ and no decoding errors are made. However, if $|N_{k,j}| > 2^{i-1}\Delta$ then a decoding error will occur. This fact can be used to try to find a bound for the error committed during the prediction process. The most straightforward method is to apply Chebyshev's inequality

$$P\left[|N_{k,j}| > 2^{i-1}\Delta\right] \leq \frac{\sigma_{N_j}^2}{\left(2^{i-1}\Delta\right)^2} \tag{1.16}$$

We adopt the realistic assumption that $N_{k,j}$ is zero-mean and variance $\sigma_{N_j}^2$. Thus, reading from Eq. (1.16), we deduce that we can take for the probability of error P_e any value greater or equal than $\frac{\sigma_{N_j}^2}{\left(2^{i-1}\Delta\right)^2}$ so that we can be sure that $P\left[|N_{k,j}| > 2^{i-1}\Delta\right]$ is fulfilled. We obviously take the equal in the inequality so that the expression of the probability of error remains as

$$P_e = \frac{\sigma_{N_j}^2}{\left(2^{i-1}\Delta\right)^2} \tag{1.17}$$

from where we can work the value of i out

$$i = \frac{1}{2}\log_2\left(\frac{\sigma_{N_j}^2}{\Delta^2 P_e}\right) + 1 \tag{1.18}$$

as the number of bits necessaries to ensure a probability of error less or equal than P_e. Note that it is not necessary to be over-conservative when choosing P_e because Chebyshev's inequality is a loose bound.

If we look thoroughly at expression (1.18) we can appreciate the presence of a new term not taken into account yet: the variance $\sigma_{N_j}^2$. This means that the data gathering node must also maintain an estimate of $\sigma_{N_j}^2$. After the initialization module (K iterations long), the data-gathering node can initialize $\sigma_{N_j}^2$ as

$$\sigma_{N_j}^2 = \frac{1}{K-1}\sum_{i=1}^{K} N_{k,j}^2 \tag{1.19}$$

However, in the main module, the variance can be computed and updated in a weighted way (filtered estimate) as

$$\sigma^2_{N_j,new} = (1-\gamma)\sigma^2_{N_j,old} + \gamma N^2_{k,j} \tag{1.20}$$

The choice of a filtered estimate is consequent with the time varying properties stated before, so we can adapt to changes in the statistics. γ is known as the "forgetting factor" and is a key value for the capacity of adapting to changes in the statistics.

Beyond our scope stands the correction/detection of errors, even though, two possible policies are suggested:

a) To detect errors, the use of a cyclic redundancy check (CRC) is proposed. In this way, each sensor would be entitled to send a CRC formed with its last m readings. Afterwards, the data-gathering node will perform its own CRC, in case both of them do not match, the data-gathering node will decide between dropping the m last readings or asking for their retransmission.

b) Another option is to use error-correction codes, as could be the case of using Reed-Solomon codes [9].

1.4 Querying and reporting algorithm

In this section, the algorithm to be implemented in the encoder and decoder is reported. Note that, at the beginning of the algorithm, the data-gathering node must gather enough information to track the correlation structure, what is achieved by performing K rounds of readings (note that K should be chosen large enough to allow the LMS algorithm convergence).

The data-gathering node should alternate the requests for "uncompressed" data among the nodes to ensure that every single node wastes the same amount of energy.

1.4.1 Data gathering node algorithm

The provided pseudocode basically expresses in programming language what has been analyzed along the previous sections. The only novelty introduced here is that, when carrying out the main module, it is necessary to ask a sensor for its full uncompressed data to keep track of the real data ensuring that the prediction is not producing erroneous estimates. However, the request for uncoded data is alternated between the different sensors, so that we can assure that the waste of energy is shared between sensors.

Once taken into account the previous considerations the pseudocode for the data-gathering node is reported:

Pseudocode for data gathering node:

Initialization:

for $(i = 0; i < K; i++)$
  for $(j = 0; j < num\_sensors; j++)$
    Ask sensor j for its uncoded reading
  end
  for each pair of values i, j
    update correlation parameters, using LMS and (1.19) equations.
  end
end

Main Loop:

for $(k = K; k < N; k++)$
  Request a sensor for uncoded reading
  for each remaining sensor
    determine number of bits, i, to request for using Eq. (1.18)
    request for i bits
  end
  Decode data for each sensor.
  Update correlation parameters for each sensor.
end

1.4.2 Sensing nodes algorithm

Pseudocode for the sensor nodes is given below.

Pseudocode for sensor nodes:

for each request
  Extract i from the request
  Get $X[n]$ from ADC
  Transmit $n \text{mod} 2^i$
end

Chapter 2

Minimum Energy Coding

2.1 Introduction

Nodes in wireless networks are usually deployed forming a very dense network in which few meters is the typical distance between them. The communication of one with each other can cause serious problems since nearby nodes can overwhelm (MAI) the received signal of the desired user. CDMA is a promising multiple access scheme for sensor and ad hoc networks due to its interference averaging properties [10]. However, the performance of CDMA systems is limited by MAI.

In the past decade numerous methods have been developed to reduce MAI, most of which focus on the design of effective correlation receivers. However, they also introduce an increase in complexity, and often, also in the demand of computational power, something which is an undesired effect. In this section a different approach focusing on source coding is studied. Instead of merely designing receivers to suppress interferences the output of the source is represented with a special codebook so that MAI is greatly reduced.

In the remaining sections we start by simply introducing the On-Off Keying modulation scheme and the ME coding to finalize with a further analysis on the system performance. This is done by first introducing the signal model of the system to subsequently analyze different relevant performance parameters.

2.2 On-Off Keying modulation scheme (OOK)

On-off keying is a basic type of modulation that represents digital data as the presence or absence of a carrier wave. In its most basic form, the presence of a carrier during the bit duration represents a binary one (i.e., 1), while a binary zero (i.e., 0) results in no signal being transmitted. This modulation technique yields not a very efficient use of the spectrum due to the abrupt changes in amplitude of the carrier wave. Having a look at its power consumption properties, however, it can be appreciated that its performance is better than that of BPSK, for instance, due to the fact that energy consumption is larger when high bits are transmitted than when low ones are. In Figure 2.1 the basic working of BPSK and OOK is depicted.

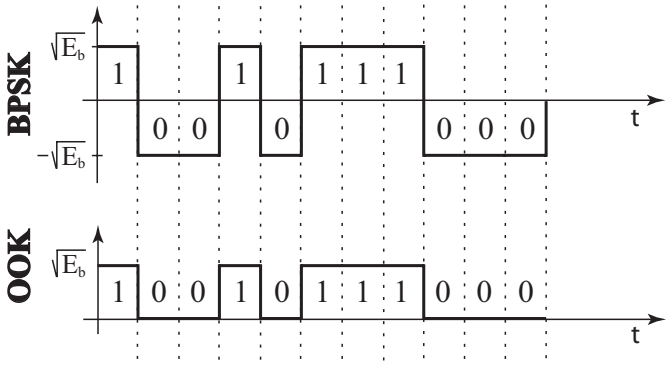

Figure 2.1: *BPSK and OOK modulation schemes.*

2.3 Energy consumption in OOK modulation

During the operation of a sensor node, power is consumed in sensing, data processing, data storing and communicating [11][12]. Among the four domains we focus in the communications one, since it is the most power consuming. The average energy consumption of a pair of nodes, one transmitting and one receiving, in a OOK modulation can be generally modelled as

[12]:

$$\begin{aligned} E_{radio} &= \widetilde{E}_{tx} + \widetilde{E}_{rx} \\ &= P_{tx,ckt}\left(T_{on,tx} + T_{startup}\right) + \alpha P_t T_{on,tx} + E_{dsp}^{(e)} \\ &\quad + P_{rx,ckt}\left(T_{on,rx} + T_{startup}\right) + E_{dsp}^{(d)} \end{aligned}$$

where $\widetilde{E}_{tx/rx}$ is the average energy consumption of a sensor node while transmitting/receiving; $P_{tx/rx,ckt}$ is the power consumption of the electronic circuits while transmitting/receiving; P_t is the output transmission power; $T_{on,tx/rx}$ is the transmitter/receiver ontime, and $T_{startup}$ is the start up time of the transceiver; $E_{dsp}^{(e/d)}$ is the energy consumed by the circuitry in encoding/decoding the data. In general, the energy spent in encoding/decoding is negligible compared to that needed to transmit and receive. Since $P_{tx/rx,ckt}$ and $T_{startup}$ are hardware dependent, these parameters cannot be used for the purpose of reducing power consumption by means of coding the source outputs. In the above expression we have also modelled the major characteristic brought about by the OOK modulation: the effective transmitting time is only a fraction of the transmitter ontime (where α is precisely that fraction of time).

2.4 The ME coding

Minimum Energy (ME) coding [13] attempts to optimize the power consumption in digital RF transmitters, which constitutes one of the most power consuming sources in portable communication devices. The function of a digital RF transmitter is to convert the modulated binary codewords into radio frequency waves able to travel through the air to reach other communication devices. The power needed to generate these signals is one of the major sources of power consumption in sensor nodes.

Any attempt to formulate the power optimization problem must be based on a deep understanding on how these waves are generated, therefore the type of modulation used is of extreme importance. For the application of ME coding, On-Off Keying (OOK) modulation is considered. Despite his limited performance when compared to other modulation schemes like

BPSK (it performs nominally 3dB worse due to reduced minimum distance in the signal constellation), the gain obtained when used along with ME is more than sufficient to justify its presence.

In Eq. (2.1) the power consumed in a WSN that uses ME coding is shown. The main characteristic of ME coding is that it reduces the value of the α coefficient (now termed α_{ME}).

$$\begin{aligned} E_{radio} &= \widetilde{E}_{tx} + \widetilde{E}_{rx} \\ &= P_{tx,ckt}\left(T^{ME}_{on,tx} + T_{startup}\right) + \alpha_{ME} P_t T^{ME}_{on,tx} + E^{(e)}_{dsp} \\ &\quad + P_{rx,ckt}\left(T^{ME}_{on,rx} + T_{startup}\right) + E^{(d)}_{dsp} \end{aligned} \tag{2.1}$$

Two key aspects should be considered now: with ME coding we are increasing the value of two system parameters, $T^{ME}_{on,tx/rx}$ and the length of the codeword L_{ME}.

- Increasing the transmitter ontime is not a disadvantage, since the major power consumption in modern low-power chips is given by P_t ($P_t >> P_{tx,ckt}$) and this term is affected by the design parameter $\alpha_{ME} << 1$. However, increasing the receiver ontime is extremely harmful because in the typical distances characteristic of a WSN, the power spent in receiving is approximately the same than that used in transmitting, so it could happen that no power savings at all are achieved. Subsection 2.8.2.2 deals with the evaluation of the ME coding power consumption properties. In order to reduce the $T_{on,rx}$ we will investigate a new coding scheme in Chapter 3.
- On the other hand, an increased codeword length could also be fatal, since we would increase the codeword error probability. However, the reduction in the multiaccess interference (provoked by the decreased number of high bits achieved by the ME coding) is more than enough to compensate this drawback.

Equation (2.1) suggests several ways of reducing the power consumption. Power consumption can be optimized by (i) minimizing $P_{tx,ckt}$ and P_t, which is done by improving the transmitter circuitry, (ii) reducing the

$T_{on,tx/rx}$, that can be done modifying the bit period, T_b (this results in an enlarged frequency spectrum) and, (iii) minimizing the presence of high bits in the codebook.

The objective of ME coding is to reduce the proportion of high bits in the codebook, α_{ME}. There are two ways of doing this:

- Use a set of codewords with a lesser number of high bits than the one used previously.
- Exploit the statistics of the source to assign codewords with a smaller number of high bits to the most probable symbols.

ME coding combines this two methods to provide the energy-optimal coding algorithm, which, based on the two previous bullets is constructed in two steps: Codebook Optimality and Coding Optimality. The former is to determine a set of codewords, termed a codebook, that has the fewest high bits, and the latter is to assign codewords having less high bits to messages with higher probability.

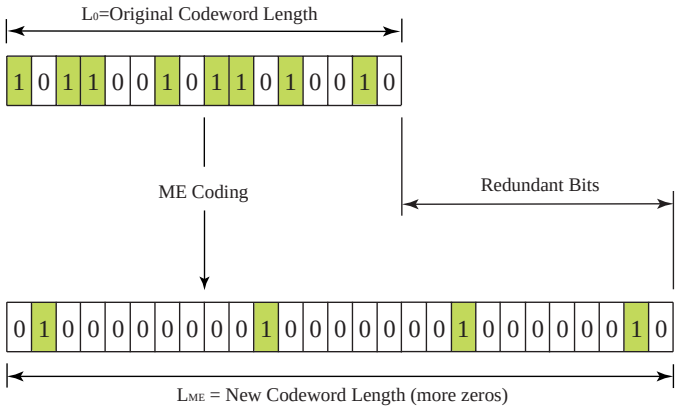

Figure 2.2: *Principle of Minimum Energy Coding.*

Basically, with ME coding we perform a mapping from an original set of codewords to another one more suitable to the aim of power savings. Each codeword in the original set has a unique image codeword in the new codebook. Specifically, the source codewords have length of L_0 bits and the ME codewords have length L_{ME} bits, with $L_{ME} > L_0$. Thus, what ME coding

does is nothing different from endowing the source codebook with new codewords having less high bits. As a result, the new codebook has larger codewords, but we do not need to use all the possible new codewords, in fact, we will just make use of those more suitable for our design goal. Figure 2.3 illustrates this concept.

Thanks to the increase of the codeword length, we will be able to allocate new codewords with a smaller number of high bits to the original codewords, reducing in this way the probability of high bit, α_{ME}, and achieving power savings by means of OOK as previously explained.

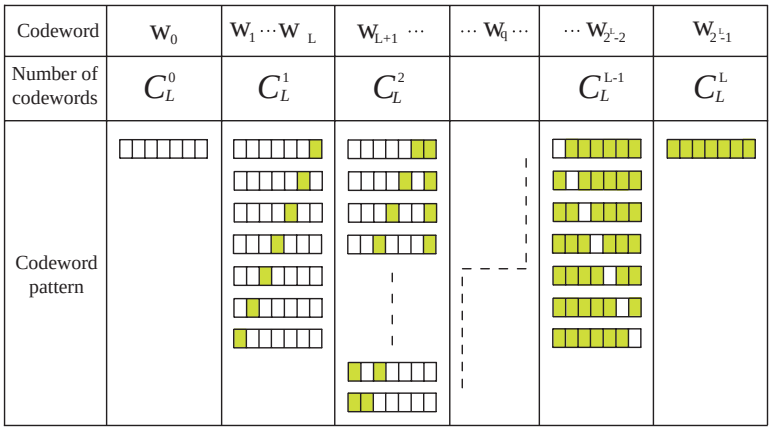

Figure 2.3: *Fixed-Length ME Codewords.*

In one sentence, ME Coding consist of: *assigning q codewords of the minimum codebook in the ascending order of number of high bits to the q-messages in the descending order of message probabilities*.

Of practical importance is ME Coding with fixed length codewords. It is clear that, for a q-codeword codebook, as the codeword length L_{ME} becomes longer, the number of high bits decreases. An extreme case is the *unary coding*, extensively studied in [14], which presents an interesting behavior. Unary coding uses a codeword length, L_{ME}, long enough to express all q messages with at most one high bit per codeword.

2.5 MAI reduction by ME source coding

The ME coding described above is useful for the reduction of MAI when using DS-CDMA. This is attained thanks to the low probability of overlap between the signals belonging to different users when we have a scenario consisting of multiple transmitters and receivers (see Figure 2.4). Since ME helps to decrease the number of high bits in the codewords it decreases the probability that two or more users transmit a high bit in the same time, achieving a reduction in the MAI.

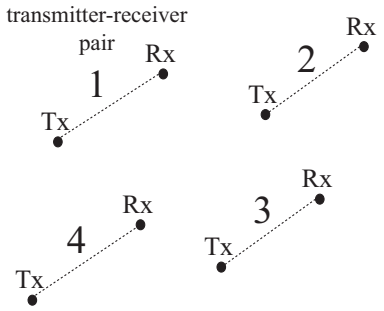

Figure 2.4: *System scenario.*

In the literature several methods for reducing MAI have been proposed. When dealing with MAI in the transmitter side, increasing the processing gain and boosting the signal are the two most common choices, but none of them are suitable for our purpose. The first one brings an undesired increase of the complexity of the receiver, while the second one moves in the opposite direction of the policy of power saving we are adopting.

Trying to reduce MAI in the receiver leads to sophisticated correlation filters that increase the complexity and cost of the design. The salient characteristic of ME coding is that the more power it saves, the smaller MAI it reaches, however, at the expense of sacrificing either transmission rate (this is not a burden in WSN where we are supposed to transmit at a low rate) or transmission time.

Figure 2.5 shows the configuration of the transmitter in our system when we use the source coding technique proposed. It is a quasi-typical DS-CDMA system in which, prior to multiplication by the spreading sequence,

we perform a mapping from the source symbols to ME codewords. At the receiver the structure is similar but inverse. We say it is a *quasi*-typical DS-CDMA system because the BPSK modulator has been substituted by an On-Off Keying.

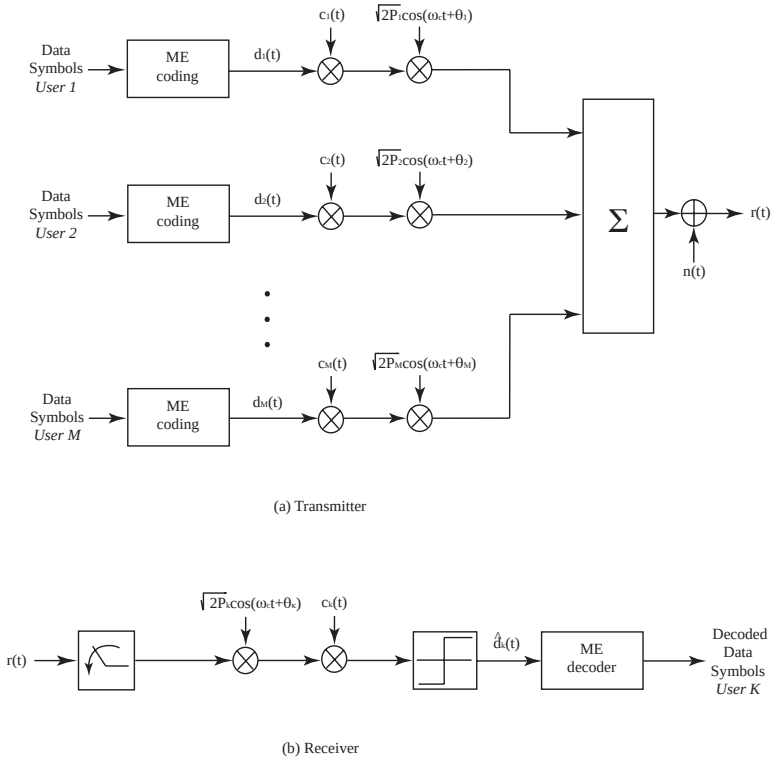

Figure 2.5: *DS-CDMA combined with ME source coding*.

The proposed scheme causes a superimposition of the signals originated in the RF transmitters and, although the DS-CDMA system could send multiple non-zero signals at the same time (unique PN sequences are assigned to different users), with the new source coding technique this chance is reduced, and hence, the performance of the system increased (due to a lowered MAI). It should be noted that as the number of wireless sensors increases, so do the interferences. This fact can be overcome by simply enlarging the codewords (individual users will have sparser non-zero signals).

2.6 Signal model

In this section we introduce the signal model of the system with which we are working. We are dealing with an asynchronous DS-CDMA system and a wireless channel where we assume we have K active transmitter-receiver pairs of nodes carrying a bit rate of $R_b = \frac{1}{T_b}$. The same fixed bandwidth W, and hence the same chip interval T_c, is allocated to each channel. Let $a_k(t)$ and $b_k(t)$ denote the spreading signal and the baseband data signal, respectively, for pair k. Each one of these signals can be expressed as:

$$a_k(t) = \sum_{l=-\infty}^{\infty} a_l^{(k)} p_{T_c}(t - lT_c) \tag{2.2}$$

$$b_k(t) = \sum_{l=-\infty}^{\infty} b_{k,l} p_{T_b}(t - lT_b) \tag{2.3}$$

Where $p_{T_c}(t)$ is a rectangular pulse which takes a constant amplitude of value one from $t = 0$ to $t = T_c$ and zero outside this interval. $p_{T_b}(t)$ is similar to $p_{T_c}(t)$ but with a time duration of T_b.

$\{a_l^{(k)}\}_{l=0}^{\infty}$ and $\{b_{k,l}\}_{l=0}^{\infty}$ are the binary sequences corresponding to the spreading sequence and baseband data signal, respectively, for user k.

In this situation, the signal at the transmitter side of the pair k can be formulated as:

$$s_k(t) = \sqrt{2P_k} b_k(t) a_k(t) \cos(2\pi f_c t + \theta_k) \tag{2.4}$$

Where P_k denotes the signal power of the transmitter node in link k, f_c is the carrier frequency and θ_k is the carrier phase.

If we now want to represent the signal at the input of the receiver in pair i, $r_i(t)$, we have to realize that it is composed by several different terms: the desired signal, the signals transmitted by other users and a noise component introduced by the AWGN channel. From now on we will consider user i, denoting it with the correspondent subscript:

$$r_i(t) = \sum_{k=1}^{K} \sqrt{2P_k\Omega_{ki}} b_k(t - \tau_k) a_k(t - \tau_k) \cos(2\pi f_c t + \psi_k) + n(t). \tag{2.5}$$

Where τ_k stands for the signal delay for the k^{th} user, $\psi_k = \theta_k - 2\pi f_c \tau_k$ and $n(t)$ is the gaussian noise with two sided spectral density $\frac{N_0}{2}$. We have also

introduced the wireless channel influence by means of the channel coefficients, Ω_{ki}, which are defined in the following way

$$\Omega_{ki} = \mathrm{PL}_{ki} e^{\xi_{ki}} \tag{2.6}$$

where PL_{ki} is the loss present in the path from the transmitter in pair k to the receiver in pair i, and ξ_{ki} is the shadow fading parameter. The path loss can be further written (in dBs) as

$$\mathrm{PL}_{ki}|_{dB} = -P_l\left(d_r\right)|_{dB} - 10n\log_{10}\left(\frac{d_{ki}}{d_r}\right)$$

with the reference distance d_r = constant = 1 meter. d_{ki} is the distance between the transmitter in channel k and the receiver in channel i, $P_l\left(d_r\right)|_{dB}$ is a known constant (55 dB for the Telos motes), and n is the path-loss decay constant, which takes the value 2 for the free space and $[3.5 - 4]$ for urban environments. The shadowing component of Ω_{ki} is described by a log-normal distribution $e^{\xi_{ki}}$, where ξ_{ki} is a zero-mean Gaussian distributed r.v. with variance $\sigma^2_{\xi_{ki}}$.

If the DS-CDMA system were completely synchronized, then one could ignore the time delays τ_k $(k = 1, 2, \cdots, K)$. Of course, this would require a perfect common timing reference for the K transmitters being necessary to introduce mechanisms for compensating the delays in the various transmission paths. However, this is not easy to implement, so that the asynchronous system is the one usually implemented.

Let us consider channel i. Since we are interested in relative phase shifts modulo 2π, there is no loss of generality in assuming $\theta_i = 0$ and $\tau_i = 0$. Following an approach similar to that found in [15] we can express the output of the correlation receiver matched to $s_i(t)$ as:

$$Z_i = \int_0^{T_b} r(t) a_i(t) \cos\left(2\pi f_c t\right) dt \tag{2.7}$$

From now on, we assume that $f_c >> T_b^{-1}$. This is a realistic assumption that helps us to ignore the double frequency component of $r(t)\cos\left(2\pi f_c t\right)$ that appears when calculating the output of the correlation receiver. Thus, it can be shown that Eq. (2.7) can be expressed as:

$$Z_i = D_i + I_i + N_g \tag{2.8}$$

where D_i is the desired signal in the channel i, I_i is the interference term due to the presence of multiple users (MAI) and N_g is a Gaussian random variable with zero mean and variance $\frac{N_0 T_b}{4}$.

$$D_i = \sqrt{\frac{P_i \Omega_{ii}}{2}} T_b b_{i,0} \tag{2.9}$$

$$I_i = \sum_{\substack{k=1 \\ k \neq i}}^{K} \sqrt{\frac{P_k \Omega_{ki}}{2}} \cos(\psi_k) B(i,k,\tau_k) \tag{2.10}$$

$$N_g = \int_0^{T_b} n(t) a_i(t) \cos(2\pi f_c t)\, dt \tag{2.11}$$

Without loss of generality we observe the output of the correlation receiver at the first time instant ($l = 0$). Also, for convenience, we have introduced the term $B(i,k,\tau)$ adopting a simplified version of that presented in [16] and used in [17].

$$B(i,k,\tau) = \int_0^{T_b} b_k(t-\tau)\, a_k(t-\tau)\, a_i(t)\, dt \tag{2.12}$$

2.7 Signal to Interference and Noise Ratio (SINR)

Once we have the signal model perfectly determined, we can undertake the task of calculating the Signal to Interference and Noise Ratio ($SINR$) parameter, which is one of the most important performance measures and can be obtained within a reasonable computational complexity. The $SINR$ for transmitter-receiver pair n (denoted as $SINR_n$) is then defined as the ratio between the average and the standard deviation of Z_i (2.13), the expectation being taken with respect to carrier phases, time delays and data symbols, but not with respect to the channel coefficients (which are supposed to be slow variant and constant for a period around T_b). However, we have to make a distinction in the computation of the $SINR$ depending on what bit is transmitted:

1. If $\mathbf{b_{i,0} = 0}$, since we do not transmit anything (recall we are using OOK), the power of the desired signal is zero, and hence, the $SINR|_{dB} = -\infty$.

2. If $\mathbf{b_{i,0}} = \mathbf{1}$, the calculation of the $SINR$ can be made via Eq. (2.13). From now on, let us consider the pair $n = i$:

$$SINR_i = \frac{E[Z_i]}{\sqrt{Var[N_g + I_i]}} \tag{2.13}$$

Since the gaussian noise and the MAI are independent, we can write:

$$Var[N_g + I_i] = Var[N_g] + Var[I_i]$$

Thus, the power of the gaussian component of the noise can be calculated as:

$$\begin{aligned} Var[N_g] &= Var\left[\int_0^{T_b} n(t)a_i(t)\cos(2\pi f_c t)\,dt\right] \\ &= \int_0^{T_b} Var[n(t)]\,a_i^2(t)\cos^2(2\pi f_c t)\,dt \\ &= \frac{N_0 T_b}{4} \end{aligned} \tag{2.14}$$

In all that follows we consider the time delays, phase shifts and data symbols $(\psi_k, \tau_k, b_{k,l})$ as mutually independent random variables. Namely, we treat ψ_k and τ_k as two r.v. uniformly distributed in their respective range of values, $[0, 2\pi]$ and $[0, T_b]$. Also, we assume that the data symbols $b_{k,l}$ take values $\{0, 1\}$ with a determinate probability $\{1 - \alpha_{ME,k}, \alpha_{ME,k}\}$. Independence among the different transmitting nodes is a realistic assumption too. All these assumptions are representative of real systems [17].

With all these hypotheses in mind, it can be proved that $E[B(i,k,\tau)] = 0$, $Var[B(i,k,\tau)] \cong \frac{2T_b^2}{3G}$ and $E\left[\cos^2(\psi_k)\right] = \frac{1}{2}$ where $G = \frac{W}{R_b} = \frac{T_b}{T_c}$ denotes the processing gain.

Therefore, assuming $\alpha_{ME,1} = \alpha_{ME,2} = \cdots = \alpha_{ME,K} = \alpha_{ME}$ and taking into account all previous considerations and Eq. (2.13), we can rewrite the $SINR_i$ as:

$$SINR_i = \frac{\sqrt{\frac{P_i \Omega_{ii}}{2}} T_b}{\left(\frac{N_0 T_b}{4} + Var[I_i]\right)^{\frac{1}{2}}} \tag{2.15}$$

The expression of the variance $Var\left[I_k\right]$ turns out to be:

$$Var\left[I_k\right] \cong \sum_{\substack{j=1 \\ j\neq k}}^{K} \alpha_{ME} P_j \Omega_{jk} \frac{T_b^2}{6G} \tag{2.16}$$

Please note the inclusion of the α_{ME} coefficient in the formula above (2.16). This is done because not all the remaining nodes may be transmitting a high bit when the user in channel i does. Taking into account that α_{ME} is the probability of sending a high bit, on average, only a α_{ME} fraction of the codeword time will be carrying ones.

If we introduce Eq. (2.16) into the $SINR_i$ we arrive at:

$$SINR_i \cong \sqrt{\frac{\frac{P_i \Omega_{ii}}{2} T_b^2}{\frac{N_0 T_b}{4} + \sum_{\substack{j=1 \\ j\neq i}}^{K} \alpha_{ME} P_j \Omega_{ji} \frac{T_b^2}{6G}}} \tag{2.17}$$

2.8 Power consumption

In this section we investigate the power consumption which takes place in the network when the ME coding is used. To start with, we solve a constrained optimization problem to subsequently evaluate the performance of the ME coding.

2.8.1 Optimal transmission power

Let us imagine a system in which we have K transmitter-receiver pairs. In this scenario (see figure 2.4), if we want to optimize the overall network power consumption, we have to consider the transmission power in each link. However, minimizing the transmission power requires to take into account the resulting effect on other system parameters as the probability of error, $SINR$...because they are all interrelated. Thus, we investigate the optimization problem:

$$\min_{\mathbf{P}} \sum_{i=1}^{K} P_i \tag{2.18}$$

$$s.t. \quad P\left[SINR_i \leq \gamma\right] \leq \bar{P}_{out} \,, \forall\, i = 1...K \tag{2.19}$$

$$P_i > 0 \quad \forall\, i = 1, \ldots, K$$

where $\mathbf{P}$ denotes the vector composed by the transmission powers of the different nodes:

$$\mathbf{P} = [P_1, \ldots, P_i, \ldots, P_K]^T \tag{2.20}$$

In order to minimize the transmission power of the overall system, we propose an optimization problem whose objective function is the sum of the powers of all the transmit nodes, while the constraints are expressed in terms of link outage probability [18].

In the optimization problem, note that γ is defined as the SINR threshold for the computation of the outage probability. In particular, the solution of the optimization problem ensures that the outage probability remains below the maximum value $\bar{P}_{out}$. Furthermore, note that the computation of the outage probability is performed with respect to the statistics of the wireless channel, and the distribution of high bits.

To solve the optimization problem (2.18), we have to model the constraints related to the outage probabilities of the links. Since the statistics of the SINR are in general unknown, we resort to the well know extended Wilkinson moment-matching method [17]. Specifically, we approximate the SINR with an overall Log-normal distribution, thus obtaining that

$$SINR_i = A_i \triangleq L_i^{-\frac{1}{2}} \approx e^{-\frac{1}{2}x_i} \tag{2.21}$$

and

$$L_i^{-1} = \frac{\frac{P_i \Omega_{ii} T_b^2}{2}}{\frac{N_0 T_b}{4} + Var[I_i]} \tag{2.22}$$

where it can be proved that $x_i \sim \mathcal{N}\left(\mu_{x_i}, \sigma_{x_i}\right)$ [17]. The approximation is useful because allows for computing the expression of the outage probability while taking into account all the relevant aspects of the wireless propagation, the transmission power, and the coding statistics. It trivially results

that

$$P\left[SINR_i \leq \gamma\right] \approx P\left[e^{-\frac{x_i}{2}} \leq \gamma\right] = Q\left(\frac{-2\ln\gamma - \mu_{x_i}}{\sigma_{x_i}}\right) \tag{2.23}$$

Let us calculate the mean and variance of A_i [19],

$$\mu_{A_i} = E_{\mathbf{\Omega}_i}\left[A_i\right] = e^{-\frac{\mu_{x_i}}{2}+\frac{\sigma_{x_i}^2}{8}} \tag{2.24}$$

$$r_{A_i} = E_{\mathbf{\Omega}_i}\left[A_i^2\right] = e^{-\mu_{x_i}+\frac{\sigma_{x_i}^2}{2}} \tag{2.25}$$

$$\sigma_{A_i}^2 = r_{A_i} - \mu_{A_i}^2 \tag{2.26}$$

It can be shown that

$$\mu_{x_i} = 2\ln M_1^i - \frac{1}{2}\ln M_2^i \tag{2.27}$$

$$\sigma_{x_i}^2 = \ln M_2^i - 2\ln M_1^i \tag{2.28}$$

where

$$M_1^i \triangleq E_{\mathbf{\Omega}_i}\left[L_i\right] \tag{2.29}$$

$$M_2^i \triangleq E_{\mathbf{\Omega}_i}\left[L_i^2\right] \tag{2.30}$$

If we recall the definition of L_i,

$$L_i = \frac{2}{P_i T_b^2 \mathrm{PL}_{ii}}\left(\sum_{\substack{j=1\\ j\neq i}}^{K} \alpha_{ME} P_j \mathrm{PL}_{ji} e^{\xi_{ji}-\xi_{ii}} \frac{T_b^2}{6G} + \frac{N_0 T_b}{4} e^{-\xi_{ii}}\right) \tag{2.31}$$

We can easily calculate the value of M_1^i and M_2^i applying the statistical expectation operator to L_i and L_i^2.

$$M_1^i = \frac{2}{P_i T_b^2 \mathrm{PL}_{ii}}\beta_1^i \tag{2.32}$$

$$M_2^i = \frac{4}{P_i^2 T_b^4 \mathrm{PL}_{ii}^2}\beta_2^i \tag{2.33}$$

where the β coefficients are given by

$$\beta_1^i = \sum_{\substack{j=1\\ j\neq i}}^{K} \alpha_{ME} P_j \mathrm{PL}_{ji} e^{\mu_{\xi_{ji}} - \mu_{\xi_{ii}} + \frac{1}{2}\left(\sigma^2_{\xi_{ji}} + \sigma^2_{\xi_{ii}}\right)} \frac{T_b^2}{6G} + \frac{N_0 T_b}{4} e^{-\mu_{\xi_{ii}} + \frac{\sigma^2_{\xi_{ii}}}{2}} \tag{2.34}$$

$$\begin{aligned}\beta_2^i = & \sum_{\substack{j=1\\ j\neq i}}^{K} \alpha_{ME}^2 P_j^2 \mathrm{PL}_{ji}^2 e^{2\left(\mu_{\xi_{ji}} - \mu_{\xi_{ii}} + \sigma^2_{\xi_{ji}} + \sigma^2_{\xi_{ii}}\right)} \frac{T_b^4}{36G^2} + \frac{N_0^2 T_b^2}{16} e^{2\left(-\mu_{\xi_{ii}} + \sigma^2_{\xi_{ii}}\right)} \\ & + \sum_{\substack{j=1\\ j\neq i}}^{K} \sum_{\substack{k=1\\ k\neq i\\ k\neq j}}^{K} \alpha_{ME}^2 P_j P_k \mathrm{PL}_{ji} \mathrm{PL}_{ki} e^{\left(\mu_{\xi_{ji}} + \mu_{\xi_{ki}} - 2\mu_{\xi_{ii}}\right) + \frac{1}{2}\left(\sigma^2_{\xi_{ji}} + \sigma^2_{\xi_{ki}} + 4\sigma^2_{\xi_{ii}}\right)} \frac{T_b^4}{36G^2} \\ & + \frac{N_0 T_b}{2} \sum_{\substack{j=1\\ j\neq i}}^{K} \alpha_{ME} P_j \mathrm{PL}_{ji} e^{\mu_{\xi_{ji}} - 2\mu_{\xi_{ii}} + \frac{1}{2}\left(\sigma^2_{\xi_{ji}} + 4\sigma^2_{\xi_{ii}}\right)} \frac{T_b^2}{6G}\end{aligned} \tag{2.35}$$

It should be noticed that neither in β_1^i nor in β_2^i there is dependence with the transmission power in channel i (P_i).

The constraint on the outage probability can be rewritten in order to evidence the dependence on the transmission power coefficients. After some algebra, a relaxation of the minimization program can be expressed as:

$$\min_{\mathbf{P}} \sum_{i=1}^{K} P_i \tag{2.36}$$

$$s.t. \quad \frac{P_i}{2T_b^{-2}\mathrm{PL}_{ii}^{-1}\left(\beta_1^i\right)^{2\left(1-Q^{-1}\left(\bar{P}_{out}\right)\right)}\left(\beta_2^i\right)^{-\frac{1}{2}+Q^{-1}\left(\bar{P}_{out}\right)}} \geq \gamma^2, \;\; i = 1 \ldots K$$

$$P_i > 0 \quad \forall\, i = 1, \ldots, K \tag{2.37}$$

The problem (2.36) is a relaxation since σ_{x_i} has been replaced with its square. This is equivalent to say that the expectations are tighter. It is possible to see that the relaxation reduces the computational burden, and that the solution is an upper bound of the solution of the original problem. The program (2.36), is a centralized problem, in the sense that to compute the solution, a central node should be able to collect all the information related to radio link coefficients, it should be able to solve the program, and finally it should broadcast the optimized powers to all other nodes. A centralize implementation exhibits clear disadvantages in terms of communication resources.

Nevertheless, it can be proved that (2.36) can be solved with a fully distributed strategy. Specifically, by following the same approach proposed in [20], each receiver node can find iteratively the optimal power as follows:

$$P_i(n) = \gamma^2 \left(2T_b^{-2} \mathrm{PL}_{ii}^{-1} \left(\beta_1^i\right)^{2\left(1-Q^{-1}\left(\bar{P}_{out}\right)\right)} \left(\beta_2^i\right)^{-\frac{1}{2}+Q^{-1}\left(\bar{P}_{out}\right)} \right)^{\{n-1\}} \quad (2.38)$$

The power updating can be done asynchronously by each node, and it can be proved that for $n \to \infty$ (n denotes time) the power converges to the optimal value [21]. The algorithm (2.38) is fully distributed, since the computation of the path loss parameter, as well as the beta coefficients, is done locally by the nodes. In particular, note hat the node i does not need to know the powers of the other nodes, but it has just to compute the expectations β_i^1 and β_i^2, through (2.32) and (2.33).

2.8.2 Numerical results

2.8.2.1 Power minimization algorithm

In this section, a numerical implementation is derived and discussed. We consider the same scenario addressed in [22] and in [18]. We consider the existence of $K = 10$ different pairs, each one with their respective transmitter and receiver. All nodes have been placed randomly within a distance of 3 to 15 meters between each other. Furthermore, we assume that $R_b = 250$ Kbps, the processing gain $G = 64$ and the path-loss decay constant $n = 4$; the power spectral density of the gaussian noise is -174 dBm, $\alpha_{ME} = 0.1$ and the expected value of the signal to interference + noise ratio threshold is set to be $E_{\mathbf{\Omega_i}}\left[SINR_i\right]_{dB} \geq 3.1$ dB ($i = 1..K$). The chosen probability of outage is the 1%.

To find suitable values for $\mu_{\xi_{ji}}$ and $\sigma^2_{\xi_{ji}}$, $i, j = 1..K$, we established a comparison between our model for the path-losses (2.6) and that found in [23][22]:

$$\Omega_{ki}|_{dB} = -P_l\left(d_r\right)|_{dB} - 10n\log_{10}\left(\frac{d_{ki}}{d_r}\right) - X_\sigma|_{dB} \quad (2.39)$$

where $n = 4$ and $X_\sigma|_{dB}$ has been shown to be a zero-mean Gaussian r.v. (in dB) with standard deviation $\sigma = 5$ representing the shadowing effects[1]. If

[1] n and σ were obtained through curve fitting of empirical data [22]

we rewrite our model (2.6) in decibels it appears to be:

$$\Omega_{ki}|_{dB} = -P_l\left(d_r\right)|_{dB} - 10n\log_{10}\left(\frac{d_{ki}}{d_r}\right) + 10\log_{10}\left(e\right)\xi_{ki} \tag{2.40}$$

Comparing both models we directly arrive at:

$$\xi_{ki} = \frac{-X_\sigma|_{dB}}{10\log_{10}\left(e\right)} \tag{2.41}$$

from where it is easy to derive

$$\mu_{\xi_{ki}} = 0 \tag{2.42}$$

$$\sigma^2_{\xi_{ki}} = \frac{\sigma^2}{\left(10\log_{10}\left(e\right)\right)^2} \tag{2.43}$$

For our simulation we assume that all links experience the same standard deviation of the slow fading (shadowing). Initially all nodes transmit at 0 dBm.

In Figure 2.6, the convergence of the limit (2.38) is shown. It can be appreciated how it barely takes 5 iterations to enter the stationary state. If we now

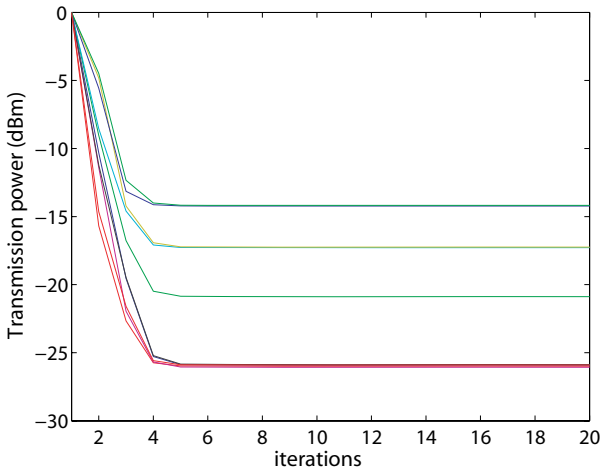

Figure 2.6: *Convergence of the power minimization algorithm.*

analyze the system we can observe that the mean value of the transmission power used in the network is -21.36 dBm and the achieved probability of outage is 0.0037, smaller than the 0.01 imposed.

2.8.2.2 Power consumption in ME coding

A comparison, in terms of total power consumption, between a typical BPSK system and a ME coding system is carried out in this subsection.

If we look thoroughly at the energy model for a system using ME coding (2.1) we can realize that, if we neglect the coding/decoding energies (as is usually done), is exactly the same than that of a BPSK system, except for the α_{ME} coefficient. We define the energy gain as the ratio of the energy used in a BPSK system and the energy used in a ME coding system.

$$\rho_{dB} = \left(\frac{E_{radio}^{BPSK}}{E_{radio}^{ME}} \right)_{dB} \tag{2.44}$$

To calculate the energy gain we have considered the CC2420 radio transceiver module by Chipcon, as is the one incorporated in the Telos motes. The values considered for the computation of the energy gain have been obtained from the CC2420 datasheet [24].

(a) $P_t = 0dBm$

α_{ME}	ρ_{dB}
0.1	2.46
0.2	2.11
0.3	1.78
0.4	1.48
0.5	1.19

(b) $P_t = -25dBm$

α_{ME}	ρ_{dB}
0.1	1.43
0.2	1.24
0.3	1.07
0.4	0.90
0.5	0.73

Table 2.1: *Energy gain of ME coding vs BPSK for two different transmission powers ($P_t = 0$ dBm and $P_t = -25$ dBm). The displayed gain corresponds to the converged value (the gain increases as the transmitting time does until it reaches a stable value).*

Two major conclusions can be drawn from Table 2.1. As we expected, the smaller the number of high bits in the ME codeword, the higher the gain. The second main result is that the higher the transmission power, the larger the gain, what is also logical, since this higher value of the transmission power allow us to further exploit the characteristics of the ME coding. This

last statement is due to the value of $P_{tx/rx,ckt}$ compared to that of P_t. Thus, we need that P_t dominates over $P_{tx/rx,ckt}$ to take advantage of the use of ME coding. In this sense, ME coding will become more and more important in the future, as advances in electronics tend to reduce the power consumption of the circuitry.

2.9 Error probability

We can express the error probability for the pair i, P_e^i, as:

$$\begin{aligned} P_e^i &= \Pr(\text{Tx } 0)\Pr(Z_i > \delta_i|0 \text{ Tx}) + \Pr(\text{Tx } 1)\Pr(Z_i < \delta_i|1 \text{ Tx}) \\ &= (1-\alpha_{ME})\, p_{e|0}^i + \alpha_{ME} \cdot p_{e|1}^i \end{aligned} \tag{2.45}$$

where δ_i is known as the decision threshold for link i and we have defined, for convenience, the probabilities given by (2.46) and (2.47). From now on, we will indicate the dependence of the parameters with the considered transmitter-receiver pair, i, with the correspondent super/subscript.

$$p_{e|0}^i = \Pr(Z_i > \delta_i|0 \text{ Tx}) \tag{2.46}$$

$$p_{e|1}^i = \Pr(Z_i < \delta_i|1 \text{ Tx}) \tag{2.47}$$

For computing these probabilities let us distinguish, once more, between the two possible cases:

1. If $\mathbf{b_{i,0}} = 0$ the output of the matched filter at the receiver is formed exclusively by the noise component:

   $$Z_{i|0} = I_i + N_g$$

   The decision variable, $Z_{i|0}$, is given by a MAI term I_i and the thermal noise N_g.

   It can be assumed that I_i can be modelled as a Gaussian random variable with a distribution that is completely specified by its mean and variance, which is, in turn, a random variable [25] due to the wireless channel coefficients. Thus, $Z_{i|0}$, as the sum of two independent

Gaussian random variables, results in another Gaussian distributed random variable.

$$\begin{array}{l} N_g \sim \mathcal{N}\left(0, \sqrt{\frac{N_0 T_b}{4}}\right) \\ I_i \sim \mathcal{N}\left(0, \sqrt{Var[I_i]}\right) \end{array} \rightarrow Z_{i|0} \sim \mathcal{N}\left(0, \sqrt{\frac{N_0 T_b}{4} + Var[I_i]}\right) \quad (2.48)$$

The previous considerations about the stochastic nature of $Z_{i|0}$, introduced by the channel coefficients, are of vital importance when calculating $p^i_{e|0}$. In fact, it is necessary to perform an average over the different realizations of the channel coefficients.

$$\begin{aligned} p^i_{e|0} &= \Pr\left(Z_{i|0} > \delta_i\right) \\ &= \Pr\left(I_i + N_g > \delta_i\right) \\ &= E_{\mathbf{\Omega}_i}\left[\Pr\left(I_i + N_g > \delta_i | \mathbf{\Omega}_i\right)\right] \end{aligned} \quad (2.49)$$

Let us calculate the probability of error for a single realization:

$$\begin{aligned} \Pr\left(I_i + N_g > \delta_i | \mathbf{\Omega}_i\right) &= \int_{\delta_i}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{Z_{i|0}}} e^{-\frac{t^2}{2\sigma^2_{Z_{i|0}}}} dt \\ &= \frac{1}{\sqrt{2\pi}} \int_{\frac{\delta_i}{\sigma_{Z_{i|0}}}}^{\infty} e^{-\frac{u^2}{2}} du \\ &= Q\left(\frac{\delta_i}{\sigma_{Z_{i|0}}}\right) \end{aligned}$$

where

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{\frac{-u^2}{2}} du \quad \text{and} \quad \sigma_{Z_{i|0}} = \sqrt{\frac{N_0 T_b}{4} + Var[I_i]}$$

finally,

$$p^i_{e|0} = E_{\mathbf{\Omega}_i}\left[Q\left(\frac{\delta_i}{\sigma_{Z_{i|0}}}\right)\right] \quad (2.50)$$

2. If $\mathbf{b_{i,0}} = 1$, there is an additional term in the output of the correlation receiver correspondent to the power emitted when the high bit is transmitted:

$$Z_{i|1} = D_i + I_i + N_g$$

Again, $Z_{i|1}$ is a random variable due to both the MAI and the desired signal components of $Z_{i|1}$. As in the case of $b_{i,0} = 0$, $Z_{i|1}$ can be modelled as a Gaussian distributed random variable.

$$Z_{i|1} \sim \mathcal{N}\left(\mu_{Z_{i|1}}, \sigma_{Z_{i|1}}\right) = \mathcal{N}\left(\sqrt{\frac{P_i \Omega_{ii}}{2}} T_b^2, \sqrt{\frac{N_0 T_b}{4} + Var[I_i]}\right) \quad (2.51)$$

Once more we can write

$$\begin{aligned} p_{e|1}^i &= \Pr\left(Z_{i|1} < \delta_i\right) \\ &= \Pr\left(D_i + I_i + N_g < \delta_i\right) \\ &= E_{\mathbf{\Omega}_i}\left[\Pr\left(D_i + I_i + N_g < \delta_i | \mathbf{\Omega}_i\right)\right] \end{aligned} \quad (2.52)$$

Let us calculate the probability of error given the channel coefficients:

$$\begin{aligned} \Pr\left(D_i + I_i + N_g < \delta_i | \mathbf{\Omega}_i\right) &= 1 - \int_{\delta_i}^{\infty} \frac{1}{\sqrt{2\pi}\sigma_{Z_{i|1}}} e^{-\frac{\left(t-\mu_{Z_{i|1}}\right)^2}{2\sigma_{Z_{i|1}}^2}} dt \\ &= 1 - \frac{1}{\sqrt{2\pi}} \int_{\frac{\delta_i - \mu_{Z_{i|1}}}{\sigma_{Z_{i|1}}}}^{\infty} e^{-\frac{u^2}{2}} du \\ &= Q\left(\frac{\mu_{Z_{i|1}} - \delta_i}{\sigma_{Z_{i|1}}}\right) \end{aligned}$$

where the variance of $Z_{i|1}$ is mainly determined by the MAI term characterized by having fast fluctuations, in contrast with the desired signal which is slow changing (we can assume Ω_{ii} approximately constant during the bit duration). One can see that $\sigma_{Z_{i|1}} = \sigma_{Z_{i|0}} = \sigma_{Z_i}$, to finally write

$$p_{e|1}^i = E_{\mathbf{\Omega}_i}\left[Q\left(\frac{\mu_{Z_{i|1}} - \delta_i}{\sigma_{Z_i}}\right)\right] \quad (2.53)$$

If we recall the average bit error probability in the system (2.45), and make use of Eqs. (2.50) and (2.53), we can write that

$$P_e^i = (1 - \alpha_{ME}) E_{\mathbf{\Omega}_i}\left[Q\left(\frac{\delta_i}{\sigma_{Z_i}}\right)\right] + \alpha_{ME} E_{\mathbf{\Omega}_i}\left[Q\left(\frac{\mu_{Z_{i|1}} - \delta_i}{\sigma_{Z_i}}\right)\right] \quad (2.54)$$

$$= E_{\mathbf{\Omega}_i}\left[(1 - \alpha_{ME}) Q\left(\frac{\delta_i}{\sigma_{Z_i}}\right) + \alpha_{ME} Q\left(\frac{\mu_{Z_{i|1}} - \delta_i}{\sigma_{Z_i}}\right)\right] \quad (2.55)$$

The probability (2.54) should be minimized with respect to the value of δ_i. Unfortunately, there is not a simple closed form solution for the value

of δ_i, due to the non linear functions involved in the computation of the expectation in (2.54). Therefore, we resort to the heuristic

$$\delta_i = \frac{\mu_{Z_{i|1}}}{2} \tag{2.56}$$

Introducing this value into Eq. (2.55) and recalling the definition for the $SINR$ we obtain

$$P_e^i = E_{\mathbf{\Omega}_i}\left[Q\left(\frac{SINR_i}{2}\right)\right] \tag{2.57}$$

To evaluate the expression above (namely, the expectation of the Q function) we will make use of the Stirling Approximation, for what we need to calculate the mean and standard deviation of the argument of the Q function. Let us start writing the Stirling approximation for the expectation of the function Q with general argument ζ^i.

$$P_e^i \approx E_{\mathbf{\Omega}_i}\left[Q\left(\zeta^i\right)\right] \approx \frac{2}{3}Q\left(\mu_{\zeta^i}\right) + \frac{1}{6}Q\left(\mu_{\zeta^i} + \sqrt{3}\sigma_{\zeta^i}\right) + \frac{1}{6}Q\left(\mu_{\zeta^i} - \sqrt{3}\sigma_{\zeta^i}\right) \tag{2.58}$$

where μ_{ζ^i} and σ_{ζ^i} are the expectation and the standard deviation of ζ^i, respectively. We have defined

$$\begin{aligned}
\zeta_1^i &= \tfrac{SINR_i}{2} \\
\mu_{\zeta_1^i} &= \tfrac{1}{2}e^{-\frac{\mu_{x_i}}{2}+\frac{\sigma_{x_i}^2}{8}} \\
r_{\zeta_1^i} &= \tfrac{1}{4}e^{-\mu_{x_i}+\frac{\sigma_{x_i}^2}{2}} \\
\sigma_{\zeta_1^i}^2 &= r_{\zeta_1^i} - \mu_{\zeta_1^i}^2
\end{aligned}$$

Finally, it should be recalled that, in an interference limited system, the real bit error probability should be computed as

$$P_b^i = \bar{P}_{out}^i + \left(1 - \bar{P}_{out}^i\right) P_e^i \tag{2.59}$$

Figure 2.7 shows the bit error probability in the network ($K = 10$ pairs). For calculating this probability the wireless channel was taken into account. The power optimization algorithm was performed so we can positively state that the probability of error obtained is the minimum achievable for the given $SINR$. The parameters used for the wireless channel and the

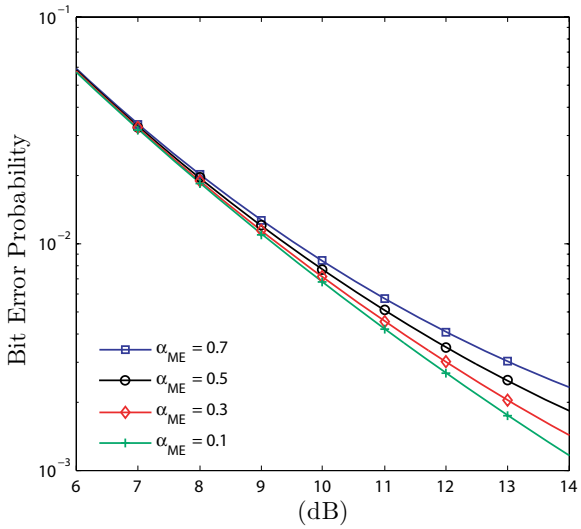

Figure 2.7: *Bit Error Probability with the variation of α_{ME} in a system using ME Coding when the wireless channel is considered and overall system power is minimized. $R_b = 250$ Kbps.*

power optimization algorithm were those presented in Subsection 2.8.2. In the x-axis we depict the average received $SINR$ (in decibels).

The main conclusion to be drawn from Figure 2.7 is that the larger the power savings (i.e, the smaller the α_{ME}), the lower the system bit error probability. This is obviously due to the fact that lower values of α decrease the multi access interference. However, ME coding is not a perfect system, and it also has undesirable effects, as are the increase in either the bandwidth requirements or in the transmission time. The former one is not a problem since bandwidth is not usually the major constraint in WSNs, but the latter could be a problem when running applications which involve the transmission of large amounts of data (which, fortunately, is not the usual case).

Chapter 3

Modified Minimum Energy Coding

3.1 Introduction

In this chapter, a modification to the ME coding is analyzed. The reason for using this new coding scheme is that in ME, due to the increased codeword length, the receiving time is also increased. However, this is an undesirable effect because in this short communications range, the power used in reception is approximately the same as that used in transmission. Thus, it may happen that the total energy consumption $E_{radio} = \widetilde{E}_{tx} + \widetilde{E}_{rx}$ is only slightly reduced, causing a reduction of the potential energy gain. To address this drawback MME was proposed [2].

3.2 The MME coding

Modified Minimum Energy coding (MME) basically consists of partitioning the ME codeword into several subframes along with the use of indicator bits. The proposed scheme is shown in Figure 3.1 and is not very different from that in Fig. 2.5. It keeps its basic structure but introduces some small changes to exploit the characteristics of MME coding.

The MME codeword is composed of several subframes each one identified by its own indicator bit. If the indicator bit is a high bit it means that there

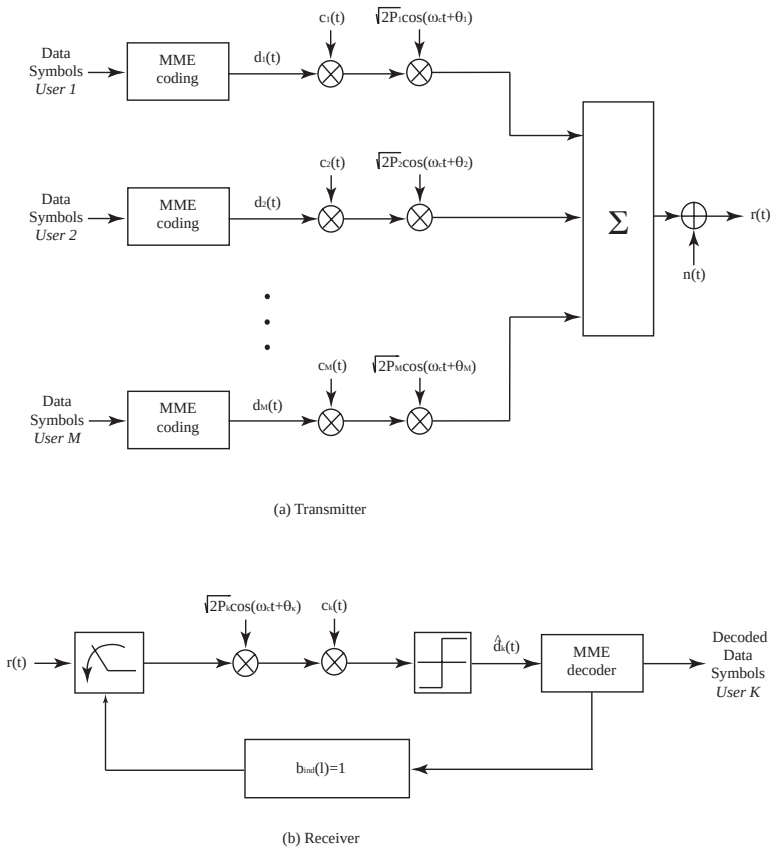

Figure 3.1: *DS-CDMA combined with MME source coding*.

are not non-zero bits in that subframe, so there is no need for decoding and we can turn off the receiver. Conversely, if the indicator bit is a low bit, it means that there are some high bits in that subframe, so decoding is compulsory. Along the chapter we present this technique and perform a thorough analysis in terms of power consumption and probability of error. The principle of MME is depicted in Figure 3.2 where the presence of the indicator bit and the subframes can be observed. As in ME coding, a fixed length codeword will be used. To be able to turn off the receiver, the system is equipped with a simple feedback loop as shown in Figure 3.1.

Both the MME coding and ME coding need to know the probability of symbol occurrence (for Coding Optimality), that is not a problem since we can assume that we are sensing application specific data which follows a determinate distribution. The contrary case in which we cannot assume a

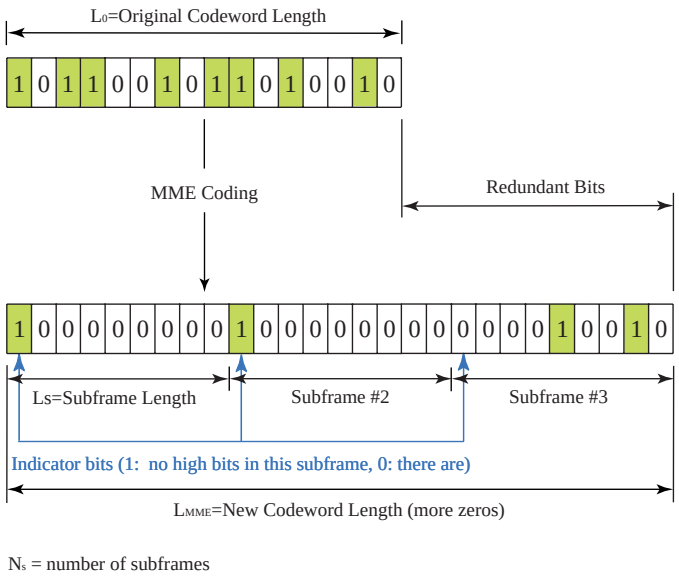

Figure 3.2: *MME coding.*

concrete distribution or we do not know the statistics of the data has been studied in [14].

An initial approach to radio energy consumption was that presented in (2.1). We will take it here also as reference, but we will introduce the characteristics brought about by the new coding scheme used. Assuming that the power needed for the startup is the same as in the normal working of the device, we can model the average consumption of radio communication as:

$$\begin{aligned} E_{radio} &= \widetilde{E}_{tx} + \widetilde{E}_{rx} \\ &= P_{tx,ckt}\left(T_{on,tx}^{MME} + T_{startup}\right) + \alpha_{MME} P_t T_{on,tx}^{MME} + E_{dsp}^{(e)} \\ &\quad + P_{rx,ckt}\left(T_{on,rx}^{MME} + (N+1)\,T_{startup}\right) + E_{dsp}^{(d)} \end{aligned} \tag{3.1}$$

where we have introduced the average number of times, N, the receiver has to awake from the down (or idle) state. The inclusion of this new term is of crucial importance when evaluating the energy consumption in the system. A more careful analysis on how energy is consumed in the receiver reveals that nowadays $P_{rx,ckt}$ is not negligible at all (actually, is the largest term between the powers), what justifies the idea that for saving energy we should reduce the time the receiver is on (on the other hand, due to the

small value of $P_{tx,ckt}$ the application of MME coding at the transmitter side does not make much sense). Unfortunately, due to the magnitude of the radio startup time, to turn the radio back on every time is needed can have a large impact on the average energy per bit. In practice it might introduce a reduction on the useful bit rates. However, for most of the applications, like medical monitoring, environment data sensing...this is not a problem, as large data rates are not required. Also, as technology evolves and smaller startup times might be reached, higher data rates can be used, allowing the use of this technique in a new broad range of applications.

In the remaining of the chapter, as it was done for ME coding, the signal model is firstly presented to subsequently present novel expressions for the evaluation of the system performance.

3.3 Signal model

Once more, let us consider an asynchronous DS-CDMA system where we have K active transmitter-receiver pairs in a local area suffering from slow fading. This system is exactly the same we already presented in Chapter 2, so all results derived there (referring to the signal model) are still valid. Thus, we just limit ourselves to present here the main results and we remit the interested reader to check section 2.6 for a more detailed analysis.

The output of the correlation receiver matched to the transmitted signal in link i is:

$$Z_i = \int_0^{T_b} r(t) a_i(t) \cos(2\pi f_c t)\, dt \tag{3.2}$$

It can be shown that Eq. (3.2) can be expressed as:

$$Z_i = D_i + I_i + N_g \tag{3.3}$$

where D_i is the desired signal in the link i, I_i is the interference term due to the presence of multiple transmitters nodes (MAI) and N_g is a Gaussian

random variable with zero mean and variance $\frac{N_0 T_b}{4}$.

$$D_i = \sqrt{\frac{P_i \Omega_{ii}}{2}} T_b b_{i,0} \tag{3.4}$$

$$I_i = \sum_{\substack{k=1 \\ k \neq i}}^{K} \sqrt{\frac{P_k \Omega_{ki}}{2}} \cos(\psi_k) B(i,k,\tau_k) \tag{3.5}$$

$$N_g = \int_0^{T_b} n(t) a_i(t) \cos(2\pi f_c t)\, dt \tag{3.6}$$

Again, without loss of generality we observe the output of the correlation receiver at the first time instant ($l = 0$). Also, for convenience, we have introduced the term $B(i,k,\tau)$ adopting a simplified version of that presented in [16] and used in [17].

$$B(i,k,\tau) = \int_0^{T_b} b_k(t-\tau)\, a_k(t-\tau)\, a_i(t)\, dt \tag{3.7}$$

3.4 Signal to Interference and Noise Ratio (SINR)

It can be demonstrated that the $SINR$ is given, as in section 2.7, by Eq. (2.17). Considering, once more, transmitter-receiver pair i

$$SINR_i \cong \sqrt{\frac{\frac{P_i \Omega_{ii}}{2} T_b^2}{\frac{N_0 T_b}{4} + \sum_{\substack{j=1 \\ j \neq i}}^{K} \alpha_{MME} P_j \Omega_{ji} \frac{T_b^2}{6G}}} \tag{3.8}$$

The only difference with the $SINR$ of a ME system lies in the α coefficient. The portion of high bits in the MME codeword (α_{MME}) is larger than that in the ME codeword due to the inclusion of the indicator bits. Say $\Pr\left(b_{ind}^{(1)}\right)$ is the probability of indicator bit equals 1 in a subframe, then α_{MME} becomes

$$\alpha_{MME} = \frac{H + \Pr\left(b_{ind}^{(1)}\right) N_s}{L} \approx \alpha_{ME} + \frac{\Pr(b_{ind}^{(1)})}{L_s} \tag{3.9}$$

where H is the number of high bits in an ME codeword. The subframe length is usually $L_s >> 1$, so the increment of α_{MME}, and hence the degradation of the $SINR$ is insignificant.

3.5 Error probability

An analysis on the system performance regarding its probability of error demands a careful study which takes into account the special nature of the MME codeword, i.e. any chosen approach should take into account the subframe per subframe decoding basis.

We will compute the average equivalent bit error probability as the ratio between the average number of erroneous bits per codeword and the codeword length. Let us denote a particular transmitter-receiver pair with the superscript i.

$$P^i_{e,MME} = \frac{\overline{n}^i_{e,sf} N_s}{L_{MME}} \tag{3.10}$$

where N_s is the number of subframes per codeword and $\overline{n}^i_{e,sf}$ is the average number of erroneous bits in a subframe transmitted in link i.

As a mean value, we can calculate $\overline{n}^i_{e,sf}$ as

$$\overline{n}^i_{e,sf} = \sum_{n=1}^{L_s-1} n p_n \tag{3.11}$$

p_n stands for the probability of having n errors in the subframe and can be computed as follows

$$\begin{aligned} p_n \;=\; & \Pr\left(b^{(1)}_{ind}\right) \Pr\left(Z_i < \delta_i \mid b^{(1)}_{ind}\right) \Pr_i \text{ (n decoding errors)} \\ & + \Pr\left(b^{(0)}_{ind}\right) \Big[\Pr\left(Z_i < \delta_i \mid b^{(0)}_{ind}\right) \Pr_i \text{ (n decoding errors)} \\ & \qquad + \Pr\left(Z_i > \delta_i \mid b^{(0)}_{ind}\right) \Pr_i \text{ (n high bits in the codeword)}\Big] \end{aligned}$$

All terms above are already familiar except for

$$\begin{aligned} \Pr_i \text{ (n decoding errors)} &= \binom{L_s - 1}{n} p_e^n \left(1 - p_e\right)^{Ls-1-n} \\ \Pr_i \text{ (n high bits in the codeword)} &= \binom{L_s - 1}{n} \alpha_{ME}^n \left(1 - \alpha_{ME}\right)^{Ls-1-n} \end{aligned}$$

where, p_e is given by (2.45) and $Q\left(x\right) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{\frac{-u^2}{2}} du$.

It should be noted here that the equivalent bit error probability computed for the MME coding system is not an exact value but a maximum bound.

This is because we did not choose any concrete codebook (for example, we considered the case in which we transmitted all ones in the codeword and we failed in receiving all of them, well, this case is simply impossible because this codeword will never be used in a MME system). Thus, if the results yielded a performance better or equal than a ME coding system, one could clearly state that in reality any MME coding system outperforms the ME coding.

Setting the system parameters equals to those selected in Section 2.9, we obtain that the probability of error is approximately the same of that achieved in a ME system. Fig. 3.3 (recall that α_{MME} can be obtained from α_{ME} (3.9)) shows the evolution of the bit error probability versus the average received $SINR$ (in decibels).

It should be recalled that, in an interference limited system, the real bit error probability should be computed as

$$P_b^i = \bar{P}_{out}^i + \left(1 - \bar{P}_{out}^i\right) P_e^i \tag{3.12}$$

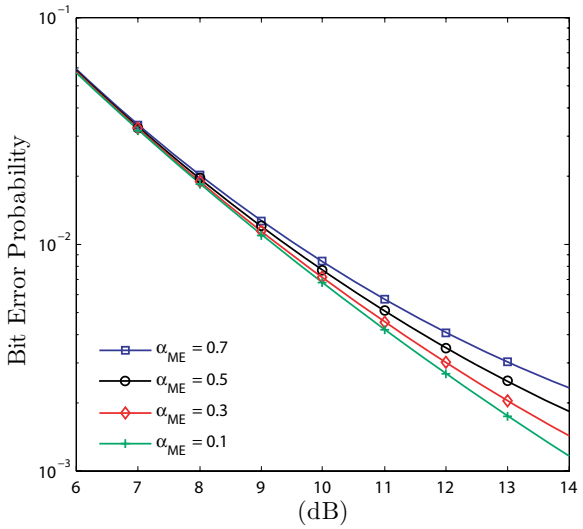

Figure 3.3: *Bit Error Probability with the variation of α_{MME} in a system using MME Coding when the wireless channel is considered and overall system power is minimized. $R_b = 250$ Kbps.*

3.6 Power consumption

In this subsection, as in 2.8.2.2, we find the expression for a comparison, in terms of total power consumed, between two systems, one using ME coding and the other MME coding.

If one assumes that the required power for CDMA decoding is approximately the same for both coding schemes (what is completely reasonable), it is easy to see that the largest power savings would come from a diminution of the receiver on-time. Let us rewrite here the expression for the energy wasted in the system (transmitter + receiver):

$$\begin{aligned} E_{radio} &= \widetilde{E}_{tx} + \widetilde{E}_{rx} \\ &= P_{tx,ckt}\left(T_{on,tx}^{MME} + T_{startup}\right) + \alpha_{MME} P_t T_{on,tx}^{MME} + E_{dsp}^{(e)} \\ &\quad + P_{rx,ckt}\left(T_{on,rx}^{MME} + (N+1)\,T_{startup}\right) + E_{dsp}^{(d)} \end{aligned}$$

MME coding aims at minimizing the energy used in the receiver so we should thoroughly analyze the effect of MME coding on the values of $T_{on,rx}^{MME}$ and N (which are the ones not hardware-dependent).

Let us start with $T_{on,rx}^{MME}$. First, it should be noted that the MME coding can increase the length of the codeword in N_s bits (where N_s is the number of subframes per codeword), with respect to that of the ME coding, due to presence of the indicator bits. Second, MME coding cuts the average number of necessary information bits to receive (per codeword) $\overline{n}_i$, (once more we consider pair i), down to:

$$\begin{aligned} \overline{n}_i &= N_s \Big[1 + \Pr\left(b_{ind}^{(0)}\right) \Pr\left(Z_i < \delta_i \mid b_{ind}^{(0)}\right)(L_s - 1) \\ &\quad + \Pr\left(b_{ind}^{(1)}\right) \Pr\left(Z_i < \delta_i \mid b_{ind}^{(1)}\right)(L_s - 1)\Big] \end{aligned}$$

Hence, the average number of bits received per MME codeword remains as:

$$\begin{aligned} \overline{n}_i &= N_s \Big[1 + \left(1 - (1 - \alpha_{MME})^{L_s - 1}\right)(1 - p_{e|0})\,(L_s - 1) \\ &\quad + (1 - \alpha_{MME})^{L_s - 1}\, p_{e|1}\,(L_s - 1)\Big] \end{aligned}$$

The total receiver ontime for receiver i (R_b is the bit rate):

$$T_{on,rx}^{i,MME} = \frac{\overline{n}_i}{R_b} \tag{3.13}$$

In second place, we have to calculate the value of N for each pair, what we will denote as N_i. This can be obtained by realizing that the radio has to be on to receive each of the indicators bits. We can assume that the radio is turned on for every subframe except for the case in which the previous subframe was decoded (the radio is already on). Mathematically, this can be expressed as:

$$\begin{aligned} N_i &= N_s \Big[1 - \Pr\left(b_{ind}^{(0)}\right) \Pr\left(Z_i < \delta_i \mid b_{ind}^{(0)}\right) \\ &\quad - \Pr\left(b_{ind}^{(1)}\right) \Pr\left(Z_i < \delta_i \mid b_{ind}^{(1)}\right)\Big] \\ &= N_s \Big[1 - \left(1 - (1-\alpha_{MME})^{L_s-1}\right)(1-p_{e|0}) \\ &\quad - (1-\alpha_{MME})^{L_s-1} p_{e|1}\Big] \end{aligned}$$

The decoding time for ME coding is $T_{on,rx}^{ME} = \frac{L_{ME}}{R_b}$, which does not take advantage of the value of α_{ME}.

We can, finally, formulate the receive energy gain as:

$$\rho_{dB} = \left(\frac{E_{radio}^{ME}}{E_{radio}^{MME}}\right)_{dB} \tag{3.14}$$

For calculating the energy gain we have considered the CC2420 radio transceiver module by Chipcon, as is the one incorporated in the Telos motes. The values considered for the computation of the energy gain have been obtained from the CC2420 datasheet [24].

In Figure 3.4, the energy gain is reported for the MME case as computed with (3.14) for different values of the α coefficient and in function of the sub-frame length. Several conclusions can be drawn from this figure:

- The smaller the number of high bits in a codeword (smaller α), the better the energy gain (no gain is achieved for $\alpha > 0.25$).
- There is an optimal value for the sub-frame length, L_s, that maximizes the energy gain. Simulations have shown that this optimal value depends on α.
- The large startup time, $T_{startup}$, characteristic of actual receivers (large compared to the bit slot) reduces the efficiency of MME coding making necessary to reduce the bit rate past 10Kbps. However, for most

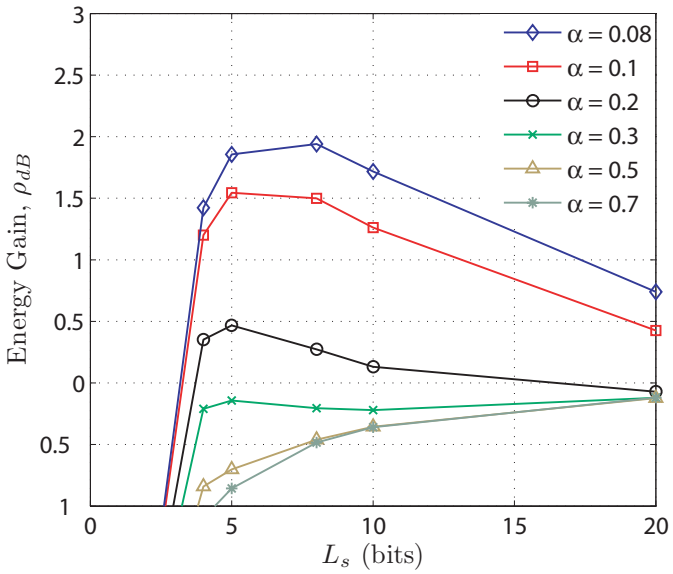

Figure 3.4: *Total Energy Gain of MME coding relative to ME coding ($R_b = 1$ Kbps, $L_{MME} = L_s N_s = 40$, $L_0 = 8$, $P_t = 0$ dBm).*

of the applications, like medical monitoring, environment data sensing...a bit rate smaller than 10Kbps is enough and gain is still achieved. What is more, as technology evolves and smaller startup times are reached, higher data rates can be used, allowing the use of MME in a broad range of exciting new applications.

Finally, it should be realized that this gain is of MME coding over ME coding. Taking into account the results achieved in subsection 2.8.2.2, we can see how for a transmission power $P_t = 0$ dBm, using a bit rate of 1Kbps and setting $\alpha_{ME} = 0.1$, $L_{MME} = 40$ and $L_s = 5$, an energy gain of 4 dB is attained (compared to a typical DS-CDMA system using BPSK).

Chapter 4

Experimental results

In this section, an implementation of the algorithm previously presented is carried out. The section is divided in three different subsections. In Section 4.1 a brief introduction to the sensors is given, followed by an explanation of the experimental setup in 4.2 to finally present the results and conclusions in Section 4.3.

4.1 Sensor description

The experiment was run onto Moteiv's popular mote: Tmote Sky. Tmote Sky is an ultra low power wireless module for use in sensor networks, monitoring applications and rapid application prototyping, being a natural replacement for Moteiv's previous product (Telos).

The module was designed to fit the size of two AA batteries from which is powered. Although 2.1 to 3.6V DC cells are explicitly requested in the datasheet ([26]) in the experiment 1.5V DC cells were used, as are the ones provided along the motes. The sensors can also be powered from the USB port of a computer (as is the case of the data-gathering node). In this case, it is not necessary to use batteries.

Also, Telos module has been designed to provide a very low power operation, for what, between many other things, uses an ultra low power Texas Instruments microcontroller (TI MSP430 F1611) featuring 10kB of RAM and

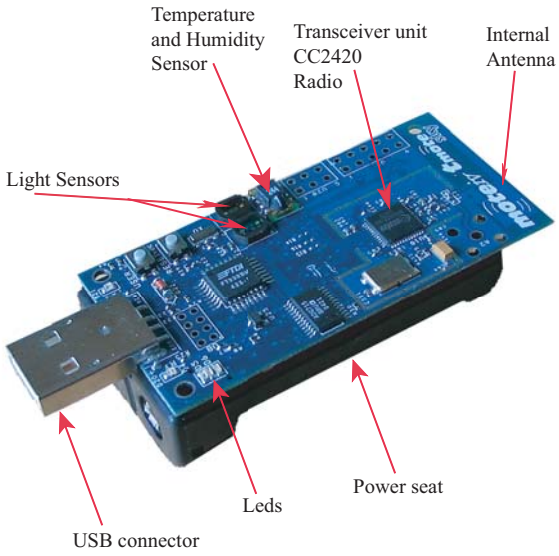

Figure 4.1: *Graphic display of the sensors used to run the experiment.*

48kB of flash. The MSP430 has an internal digitally controlled oscillator (DCO) that may operate up to 8MHz.

To be able to communicate with a PC through the USB port, Telos uses a USB controller from FTDI which, of course, requires a previous installation of FTDI's drivers on the host. Furthermore, Windows users will need the Virtual Com Port (VCP) drivers. These drivers are included on the Moteiv CD shipped with your order or downloaded from FTDI's website.

On the radio interface, Telos features the Chipcon CC2420 radio for wireless communications. The CC2420 is an IEEE 802.15.4 compliant radio, being highly configurable. Two antennas options are provided, an internal antenna built into the module and an external SMA connector for connecting to external antennas. By default, Telos is shipped with the internal antenna enabled. Although not a perfect omnidirectional pattern, the antenna may attain 50-meter range indoors and upwards of 125-meter range outdoors.

There are 4 optional sensors supported onboard:

- The TSR (*Total Solar Radiation*) and PAR (*Photosynthetically Active Radiation*) sensors measure using the microcontroller's 12-bit ADC with $Vref = 1.5V$. The photodiodes create a current through a $100k\Omega$ resistor. To calculate this current we can use Ohm's Law:

$$I = V_{sensor}/100k\Omega \tag{4.1}$$

  where V_{sensor} can be obtained as:

$$V_{sensor} = value_{ADC}/4096 \cdot Vref \tag{4.2}$$

  The Moteiv datasheet [26] includes curves for converting the photodiode's current into light values (Lux)

- Humidity and Temperature sensors are located in the external Sensirion sensor. Their readings can be converted to IS units as follows:

  For temperature, the 14 bits value returned can be converted to Celsius degrees as:

$$temperature = -39.60 + 0.01SOt \tag{4.3}$$

  where SOt is the raw output of the sensor.

  Humidity is a 12-bit value that is not temperature compensated.

$$humidity = -4 + 0.0405SOrh + (-2.8 \cdot 10^{-6})(SOrh^2) \tag{4.4}$$

  where, same as before, $SOrh$ is the raw output of the relative humidity sensor. Using this calculation and the temperature measurement, you can correct the humidity measurement with temperature compensation:

$$humidity_{true} = (Tc - 25)(0.01 + 0.00008SOrh) + humidity \tag{4.5}$$

  where Tc is the temperature measured in Celsius from equation (4.3), $SOrh$ is the raw output of the relative humidity sensor, and humidity is the uncompensated value calculated in equation (4.4).

4.2 Experimental setup

Several important variables have to be defined before we are able to run the experiment of distributed source coding.

The first one is to decide how the WSN is deployed. Is straightforward to choose the network architecture, since it is requested by the algorithm itself to be as shown in Fig: 1.1. Furthermore, we choose the network to be composed by five sensors to be able to compare results with those presented in [3].

Since the data-gathering node is represented by a sensor plugged in the USB port of the PC where the algorithm is run, we have now to decide which interface are we going to choose for reading and writing to the USB port and perform the necessary calculations. Here we have several possible solutions being the most typical ones to write a suitable code in Java or C. However, we decided to interface the motes with MATLAB. The main reason being that MATLAB is a well known platform providing a complete support for matrix processing and where we can reuse code for subsequent simulations (see section: 4.4). We have provided a description on how to interface MATLAB with TinyOS in Appendix B.

The experiment took place in the laboratory of the Automatic Control Group (School of Electrical Engineering) at KTH. In Fig. 4.2 the location of each sensor is shown. Each star denotes the position of the sensor with network address the displayed number. Our application has been developed on top of a multihop protocol so that direct line of sight is not required. The routing protocol will build the routing tree having node with network address zero as root, what means that all packets will be forwarded to this node. Hence, the node zero (we will denote the sensors by their network addresses) will be the one in charge for requesting data to a concrete node and receiving and processing the correspondent answer. The computer to which the Base Station (BS) is attached, will be the one responsible for tracking the correlation structure and determining when and which sensor must be enquired.

In the experiment we will test the behavior of the algorithm subject to two

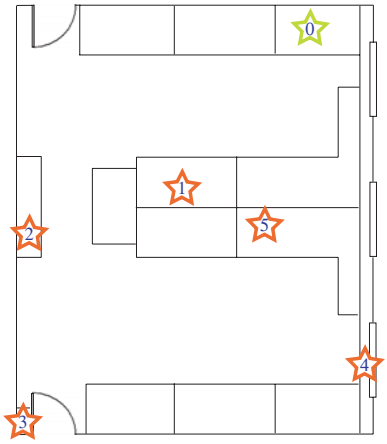

Figure 4.2: *Physical disposition of the sensors in the lab.*

different environments: indoor and outdoors. The latter one was achieved by opening the windows.

When computing the prediction for a determinate sensor we use its most recent four past measures and the current value sensed by one of the remaining nodes. Mathematically, this is expressed as

$$Y_k^{(j)} = \sum_{l=1}^{4} \alpha_l X_{k-l}^{(j)} + X_k^{(m)} \tag{4.6}$$

where, obviously, $m \neq j$

To perform the experiment we chose the temperature as the magnitude to be measured. From Section 4.1 we know that the ADC returns a 14 bit value, and from Eq. (4.3) we derive that the dynamic range expands over $[-39.60, 124.23]$ ºC. But we still have to come up with the value of several important variables as the step size, the value of K (length of the initialization module), the sample time, the maximum waiting time (the time we wait after a request for concluding whether a packet has been lost or not),...

Let us start by the step size μ and K (see Eqs. (1.13)(1.19)), since they are closely related to each other. The value of μ and K will be given by the initial conditions of the coefficients' vector $\mathbf{\Gamma}_j$ and the characteristic time of the signals sensed. Several simulations over real data yielded $\mu = 2.1 \cdot 10^{-4}$ as the quasi-optimal tradeoff value between speed of convergence and

stationary error (following the original theory by Haykin [27] we initialize $\mathbf{\Gamma}_j$ as the null vector). For K the value chosen was 30.

As sample time we chose 10 seconds (between measures belonging to one self-same sensor), in other words, two sensors with consecutive network addresses are enquired with a time difference of 2 seconds.

Finally, we set the value for the waiting timer to 3 seconds and the probability of decoding error to 0.01.

4.3 Analysis of the results

With the parameters defined in the previous section, we ran the experiment during approximately one hour and a half to yield the results shown in Figs. 4.3, 4.4 and 4.5. Let us give a small insight on each one of the subplots:

- Fig. 4.3 plots the value of the sensed data and its predictions during the run of the experiment. Note that signals are perfectly tracked, committing an unnoticeable error, so that we can only appreciate 5 different signals.

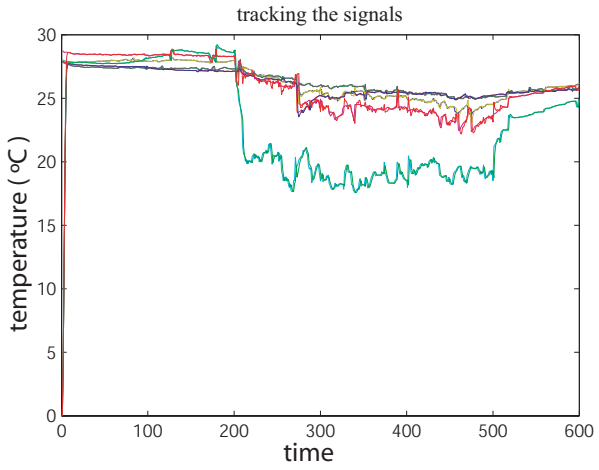

Figure 4.3: *Signals' reconstruction.*

- Fig. 4.4 shows the evolution of the error (difference between the decoded and the predicted value of the temperature) during the initialization module. It can be appreciated how it barely takes 8 samples to

track the signals with an error smaller than 0.05ºC what validates the value chosen for the step size μ, and shows that we can decrease the length of the initialization module (in other words, reduce K), with the consequent increase of the compression rate.

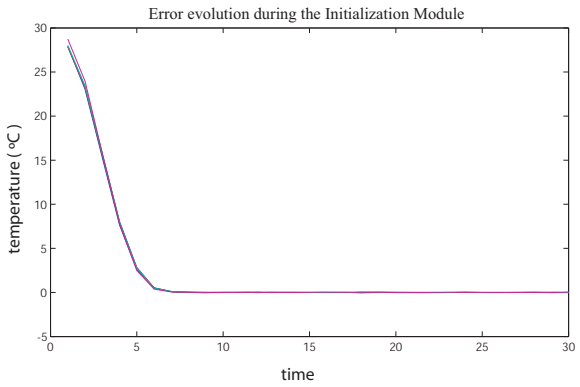

Figure 4.4: *Error evolution.*

- Fig. 4.5 is the most illustrative one when trying to show the reduction achieved in the number of bits requested by means of the algorithm. In the y-axis it shows the frequency with which a determinate number of bits (in the x-axis) has been retrieved. We can realize how, without any compression, this plot would be a single bar at value $i = 14$ with frequency 1. However, thanks to the compression algorithm we have been able to displace it to its right, being the new median of the distribution around value $i = 6$ (bits) for most of the sensors.

Now that we have a basic knowledge of what each figure means, it is mandatory to make a joint analysis of the three figures without which the potential of the compression algorithm would not be understood.

Initially the sink, where the prediction algorithm is carried out does not have any information to compute the prediction. This is the reason why at the beginning the difference between the real and the predicted values is so large (see Fig. 4.4). As the BS starts to collect measurements from several nodes, predictions become to be more and more accurate until they reach an almost perfect estimate. Once the signals have been tracked, the number of requested bits starts to decrease because the variance of the error

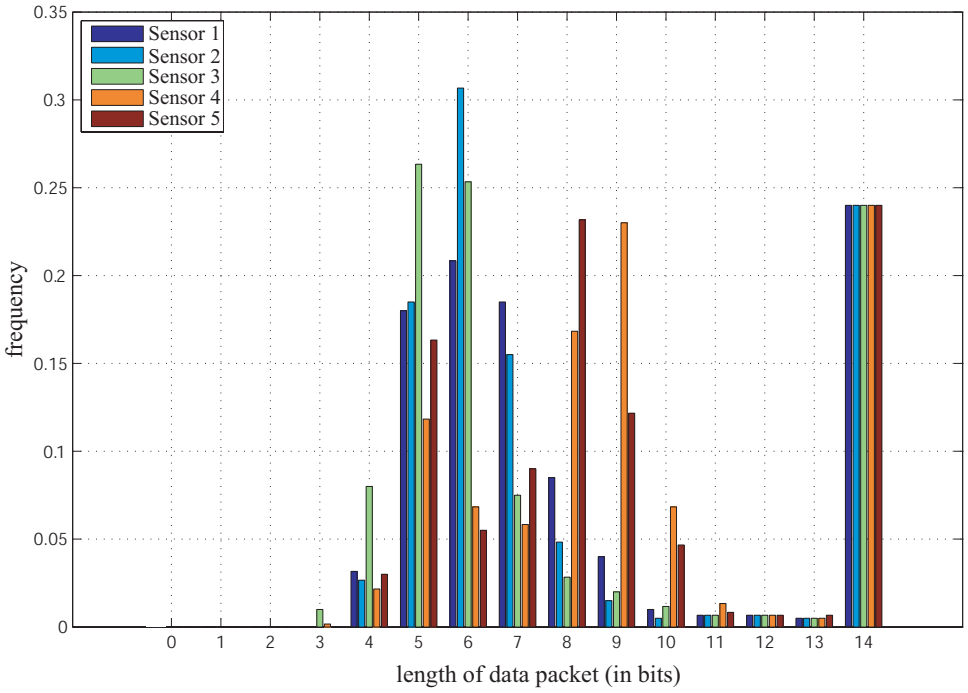

Figure 4.5: *Length of the data packets.*

decreases as a consequence of the smaller prediction error (see Eq. 1.20). In the experiment two kind of situations can be seen: the first one corresponds to an indoor environment (for all the sensors from samples K=1:200), in that moment, the window close to sensor 4 was opened (see Fig: 4.2 for more information about the location of the sensors) so that the temperature returned by this sensor (and number 5) reflects a decrease of its value at the same time that the variance becomes higher, as is typical of an outdoor environment. This can be easily appreciated in Fig. 4.3. We can see how one sensor drastically decreases its temperature at the same time the rest of sensors start to slowly decrease their sensed values of temperatures little by little(due to the slow cooling of the room). In $K = 500$ the window was closed again with the consequent stabilization and slow increase of the temperature in the room.

In Table 4.1 we show the compression rate achieved during different modules of the experiment. Note that in the table, IM stands for Initiation Mod-

ule (see Section)

Description	Case	Compression rate (%)
Whole experiment excluding IM	$k = K..600$	37.8
Whole experiment including IM	$k = 0..600$	36
All windows closed, no IM	$k = K..200$	40.08
One window opened	$k = 200..500$	34.76

Table 4.1: *Analysis of the compression rate achieved.*

Understanding Table 4.1 is of crucial importance for a deeper comprehension of the algorithm. Hence, we would like to highlight some aspects:

1. Including the Initialization Module in the computation yields a reduction in the compression rate achieved. In this way, a smaller value of K (it can be set to 10 instead of 30) will perform better.

2. Stationarity of the signals yields more precise estimates. Thus, the compression rate obtained when the windows are closed is higher than that got when one window is open. This is easily understandable: the colder external temperature provokes a descend of the temperature in the room. This change in the statistics of the temperature has as a consequence a deviation between the real value and the predictions (increase of the error), whose ultimate consequence is the increased number of bits requested.

4.4 Results of the simulation

In this section further studies on several parameters of the algorithm are presented. The objective of carrying out this simulation work is threefold: *i)* Overcome the always cumbersome and time consuming work of setting up the network, *ii)* arrive at results comparable to those of the experiment (which is unique and unrepeatable) by using the data obtained from this one, *iii)* perform analysis impossible to carry out in reality.

Simulations are performed in a similar way to that of the real experiment. The structure of the MATLAB code is basically the same as in the experiment but instead of requesting data to the correspondent node, it reads the appropriate variable in a log file. Since the data used was recorded in the experiment and we can assume that it faithfully represents the reality (the decoding error is null as it will be shown in subsection 4.4.2), we can consider this data as real.

Prior to going farther we should check the validity of the results given by the simulator. To do this we just need to introduce the measures recorded in the experiment into the simulator, which returns a compression ratio of a 36%, value which fits in that attained in the real implementation.

Once the use of simulations has been motivated and validated, we can proceed to study the effect of several parameters on the compression rate and the robustness to errors of the compression algorithm.

4.4.1 Effect of K and the number of sensors

Let us start our analysis with a parameter which obviously affects the overall compression rate: the length of the initialization module, K.

During the initialization module (IM) the sensors are asked to send their data uncompressed. Thus, while IM is taking place, compression is not being carried out. In Table 4.2 we show the compression rate achieved for three different runs.

K	Compression rate (%)
50	34.5
30	36
10	37.4

Table 4.2: *Varying the value of K.*

Simulations confirm what we already knew: the higher the value of K, the smaller the compression rate. It should also be noted that the influence of K in the final compression gain also depends on the relative value of K

respect to the length of the simulation run (this can be appreciated in Table 4.3 if we consider one individual row).

An analysis of how the number of nodes affects the overall performance was also carried out. The results have been displayed in Table 4.3, showing that larger networks attain a better performance than smaller ones. Thus, if we compare the compression gain for a WSN composed by 5 sensors and another one having 200 nodes, we can easily see an improvement in the compression gain of almost the 25%.

	number of measures				
number of sensors	600	5000	10000	30000	100000
5	37.37	37.93	37.97	37.99	38
20	44.38	45.04	45.09	45.12	45.13
50	45.78	46.47	46.51	46.54	46.55
100	46.25	46.94	46.98	47.02	47.03
200	46.48	47.18	47.22	47.26	47.27

Table 4.3: *Optimizing the compression gain.*

In Table 4.3, we assumed $K = 10$, and the IM was included for the calculation of the compression gain. For computing the presented values we considered the same distribution for the number of requested bits as the one in the experiment.

4.4.2 Robustness to errors

In this section we check the robustness of the compression algorithm against the two kinds of errors that can appear. The first type of error is a packet loss. The second is a decoding error, which means that we decoded the prediction to the erroneous codeword in the codebook. Each one of these errors will be considered in the following items:

Packet loss There are two possible ways in which a packet loss can occur: malfunction of the temperature sensing device or transmission loss. In principle, a packet loss could enormously affect the decoding

and the correlation tracking processes, because this error propagates along time (affecting those estimates that depend on this measurement). There are two possible policies to follow: use the prediction on behalf of the sensed value or simply use the last correctly decoded value for that sensor. We chose the latter one. For the simulations we considered a bursty noise channel: the Gilbert-Elliott Channel model [28], which is characterized by two states, the *Good* and the *Bad* state, denoted by *G* and *B*. Let us express the probabilities of being in each of these states as π_G and π_B. The wireless channel is modelled by choosing the model parameters to match a concrete probability of packet loss and the average burst length. The Gilbert-Elliott model has been depicted in Fig. 4.6, where p_{ij} $(i, j \in \{g, b\})$ is the probability of moving from state i to j, and p_{ii} the probability of remaining in state i if the previous state was also i.

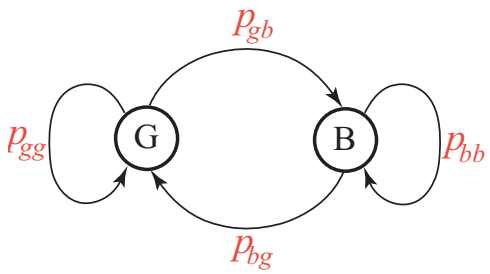

Figure 4.6: *Gilbert-Elliott channel model.*

From basic probability theory:

$$p_{gb} + p_{gg} = 1 \tag{4.7}$$

$$p_{bb} + p_{bg} = 1 \tag{4.8}$$

The Gilbert-Elliott Model is a first order, 2-state Hidden Markov Model, thus we can write the transition matrix, M, as:

$$M = \begin{bmatrix} p_{gg} & p_{gb} \\ p_{bg} & p_{bb} \end{bmatrix} \tag{4.9}$$

which, by using Eqs. (4.7)(4.8), can be rewritten as:

$$M = \begin{bmatrix} p_{gg} & p_{gb} \\ p_{bg} & p_{bb} \end{bmatrix} = \begin{bmatrix} 1-p_{gb} & p_{gb} \\ p_{bg} & 1-p_{bg} \end{bmatrix} \tag{4.10}$$

However, before proceeding, we have to find the relation between the variables of our channel (average burst length, $\bar{l}$, and probability of packet loss, p_{pl}) and the variables of the Gilbert-Elliott model (p_{bg} and p_{gb}).

The first relationship is easy to calculate and can be shown to be:

$$\bar{l} = \frac{1}{p_{bg}} \tag{4.11}$$

Calculating the second relationship is a little bit more laborious. Let us start by writing the local balance equations along with the property that the sum of all state probabilities has to be one:

$$p_{gb}\pi_G - p_{bg}\pi_B = 0 \tag{4.12}$$

$$\pi_G + \pi_B = 1 \tag{4.13}$$

It is immediate to verify that:

$$p_{gb} = \frac{p_{bg}\pi_B}{1-\pi_B} \tag{4.14}$$

Finally, by making use of eqs. (4.11) and (4.14) our model remains totally determined by the parameters of the real channel:

$$\begin{aligned} p_{bg} &= \frac{1}{\bar{l}} \\ p_{gb} &= \frac{\pi_B}{\bar{l}\,(1-\pi_B)} \end{aligned}$$

Where we would like to highlight that the probability of packet loss is the probability of being in the *bad* state, $p_{pl} = \pi_B$.

First of all, we should check if the simulated channel effectively corresponds to the one described by the design parameters. Thus, we ran the simulations several times under different channel parameters and analyzed the resulting systems. The output of these analysis are shown in Table 4.4.

Desired Channel		Simulated Channel	
π_B	$\bar{l}$	π_B	$\bar{l}$
0.1	11	0.099	10.643
0.2	5	0.199	4.885
0.3	8	0.302	8.089

Table 4.4: *Testing the channel.*

The main reason for introducing the channel was to check the robustness to packet losses of the compression algorithm. In Fig. 4.7 we can graphically see the destructive effect of the bursts of errors. There are two bursts spreading over the time ranges $[474 - 493]$ and $[503 - 518]$ where the reception of packets is interrupted (it can be deduced that we are dealing with bursts from the fact that the decoded data remain constant, not being able to follow the evolution of the real data). As it was programmed, we stick to the last correctly received temperature measure, committing decoding errors during these ranges. To study the performance of the algorithm in terms of probability of decoding error, a set of simulations were carried out, the results being reported in Table 4.5.

	Average Burst Length				
Packet Loss Rate	1	3	5	10	15
0 %	0	0	0	0	0
10 %	$7.33 \cdot 10^{-2}$	$7.5 \cdot 10^{-2}$	$7.83 \cdot 10^{-2}$	$8.12 \cdot 10^{-2}$	$9.45 \cdot 10^{-2}$
20 %	$1.56 \cdot 10^{-1}$	$1.58 \cdot 10^{-1}$	$1.81 \cdot 10^{-1}$	$1.85 \cdot 10^{-1}$	$1.95 \cdot 10^{-1}$
30 %	$2.42 \cdot 10^{-1}$	$2.71 \cdot 10^{-1}$	$2.74 \cdot 10^{-1}$	$2.81 \cdot 10^{-1}$	$2.87 \cdot 10^{-1}$

Table 4.5: *Probability of decoding error for several experimental setups.*

Prior to drawing any conclusions, it should be noticed that we are superimposing the probability of packet losses to that of making a decoding error due to the prediction and decoding algorithm (recall that this probability was set to 0.01). Having this in mind, there are three major results to be highlighted from Table 4.5:

a) The first one is to notice that for a probability of having zero packet

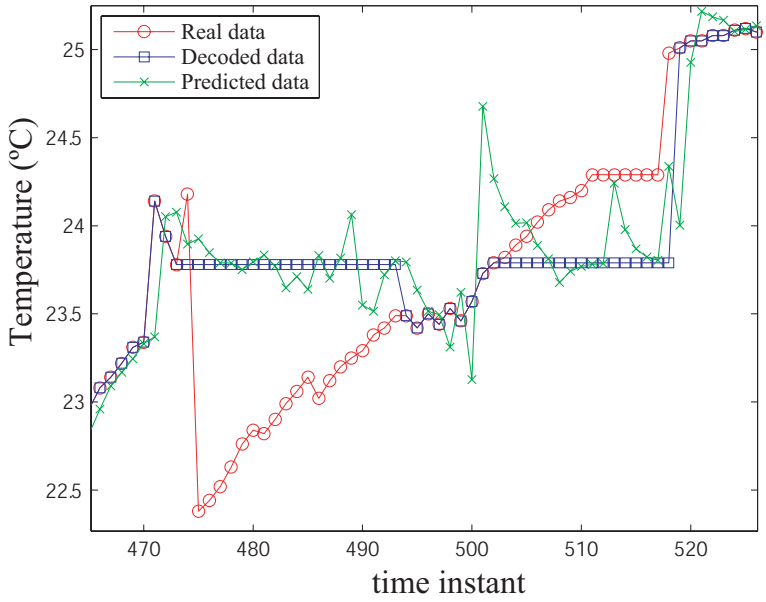

Figure 4.7: *Effect of the burst noisy channel.*

losses, the algorithm is able to perform without doing any errors;

b) The second thing to highlight is that the longer the bursts, the larger is the probability of decoding errors;

c) The third thing to remark is that for every simulation carried out, the achieved probability of decoding error was smaller that the actual packet losses;

From the previous observations we can conclude that the algorithm is robust to packet loss.

Decoding error Anytime the decoded measure does not match the real measure we say to have a decoding error. Recall we set the probability of decoding error to be $P_e = 0.01$, and that it was a basic parameter for choosing the number of bits we wanted to receive from the sensors (1.18) along with the use of Chebyshev's bound. It was also stated that Chebyshev's inequality was a too loose theoretic bound. The purpose of this paragraph is to experimentally motivate this statement. In this sense, a comparison between the tolerable noise and the prediction noise was carried out. By tolerable noise we mean the amount of noise that can exist between the prediction of a sensor

reading and the actual sensor reading without inducing a decoding error, and is calculated as $2^{i-1}\Delta$, where i is the number of requested bits and Δ is the quantization step. On the other hand, with prediction noise we just denote the difference between the value of the estimate and the actual sensor reading. A plot of the tolerable prediction noise versus the actual prediction noise is given in Fig. 4.8.

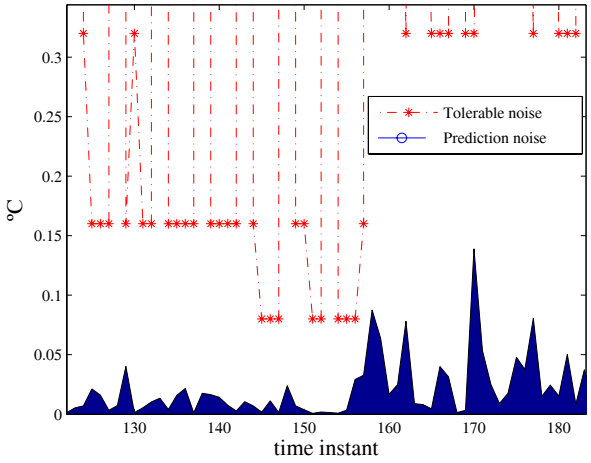

Figure 4.8: *Tolerable versus prediction noise.*

From Fig. 4.8 we can conclude that the tolerable noise is much higher than the actual prediction noise. This is because we chose a non-aggressive policy when determining the number of necessary bits to request data to the sensors. If we choose a more aggressive policy, we will be able to achieve an improved compression gain, but we will also get closer to the probability of error we set. For example, for a $P_e = 0.01$ we propose to use the following heuristic formula to calculate the number of bits, instead of using (1.18).

$$i = \frac{1}{2}\log_2\left(\frac{\sigma^2_{N_j}}{\Delta^2 P_e}\right) + 0.1 \tag{4.15}$$

If we simulate once again with this new formula, we obtain a compression gain of a 42% and a probability of error equals $9.67 \cdot 10^{-3}$, what clearly outperforms the results drawn when we used (1.18). If

we plot again (Fig. 4.9) the tolerable noise versus the prediction noise we can see how our margin has been reduced below the quantization step.

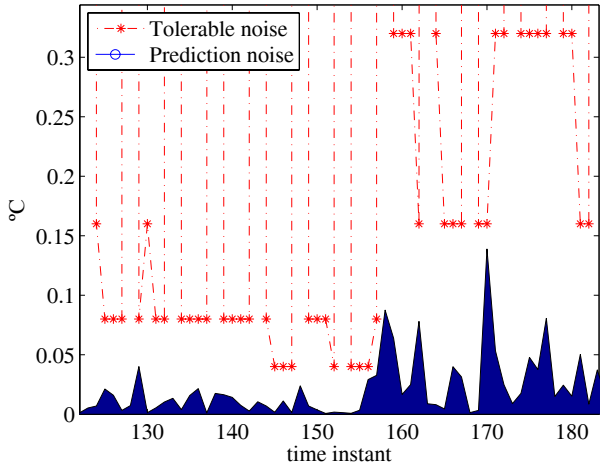

Figure 4.9: *Tolerable versus prediction noise with improved number of requested bits, i.*

Chapter 5

Conclusions

Wireless sensor networks are more than just a specific form of ad hoc networks. The stringent miniaturization and cost requirements make economic usage of energy and computational power a significantly larger issue than in normal ad hoc networks.

On the distributed and adaptive source coding, we implemented the algorithm analyzed in Chapter 1, yielding an almost perfect tracking of the environmental magnitudes being sensed. A compression gain of around 50% was shown to be achievable for large WSNs and enough number of samples. Furthermore, the algorithm was shown to be robust to packet losses through a simulation of a Gilbert-Elliott channel.

In the chapters of this thesis, different joint source-channel coding schemes to reducing power consumption have been exposed and analyzed. Firstly, we have dealt with ME coding and showed how a reduction of the MAI can be achieved. ME coding allows to save energy when using a DS-CDMA scheme in which the usual BPSK modulation is substituted by an OOK one, which takes advantage of the large run of zeros. However, an enlarged codeword may exhibit two major drawbacks: on the one hand it increases the bit error probability, on the other it increases the power used on the receiver.

To address this problem MME coding has been analyzed and compared to ME coding in terms of relative gain. Novel expressions for the evaluation of the bit error probability and the power gain have been provided.

We have also proposed and solved a constrained minimization problem to adapt the optimal transmission powers for the nodes in the WSN, so that the consumed power in the network is the smallest possible while satisfying a given constraint.

Appendix A

Source Coding Basics

A.1 Introduction

Communication systems are designed to transmit information from sources to destinations. Sources of information can be analog or discrete. An example of the former case can be a phone call, where generally, the source is an audio signal. The output of this source is analog and, hence, they are called "analog sources". On the other hand, we have "discrete sources" which have discrete values as outputs, as can be the daily stock market index, computer files...

Whether the source is analog or discrete, a digital communication system is designed to transmit information, consequently, the output of a source must be conveniently treated to be transmitted digitally. This function is performed by the source encoder at the transmitter side, whose task is not only to quantize the signal, but also to make an efficient representation of the information in digital form (note we haven't established yet what we mean by efficient).

Obviously, the smaller number of bits is used to represent the signal (during quantization), the less capacity of the channel is used, but the worse fidelity is achieved. In general, we will permit a given level of distortion in quantization and we will try to make the most efficient representation of the levels the signal can take.

A.2 Mathematical models for information sources

Any information source has an output that can be described statistically. Indeed, if we knew the output of a source in advance (deterministic output), there would be no need to transmit it.

Therefore, we need now to model each of the sources described before:

- The output of a discrete source is a sequence of letters belonging to a finite alphabet of L possible letters: $x_i \in \{x_1, x_2 ... x_L\}$.
  Each letter has a probability of occurrence:

$$p_k = P(x = x_k) \tag{A.1}$$

$$\sum_{k_1}^{L} p_k = 1 \tag{A.2}$$

  If every letter satisfies statistical independence among all past and future outputs, we say it is a discrete memoryless source (DMS). On the other hand, if the discrete output shows statistical dependence, we should construct a mathematical model which fits this dependence. For instance, a discrete source is said to be stationary if the joint probabilities of two sequences of length n, say $a_1, a_2, ..., a_n$ and $a_{1+m}, a_{2+m}, a_{n+m}$ are identical for $n \geq 1$ and all possible shifts of m.

- An analog source can be modelled as one whose output presents a waveform $x(t)$, which is a sample function of a stochastic process $X(t)$. We assume that $X(t)$ is stationary with correlation $\phi_{XX}(\tau)$ and power spectral density $\Phi_{XX}(f)$ and bandlimited,$\Phi_{XX}(f) = 0 \; \forall \; |f| \geq B$. The sampling theorem helps us to transmit the samples of the analog signal ($x_n = x(n) = x(nT_s) = x(n\frac{1}{2B})$) for further reconstruction at the receiver side as:

$$X(t) = \sum_{n=-\infty}^{\infty} X(\frac{n}{2B}) Sa(2\pi B(t - \frac{n}{2B})) \tag{A.3}$$

  Where $Sa(x) = \frac{sin(x)}{x}$.

  Our previous analog signal is now a discrete-time signal.

Finally, we should observe that the result of sampling an analog source is usually a discrete-time continuous-amplitude signal, being necessary to perform quantization to obtain a digital signal.

Where $\{X(\frac{n}{2B})\}$ denote the samples of the process $X(t)$ taken at the Nyquist rate ($f_s = 2B$ samples/s).

A.3 A logarithmic measure of information

Now we know the information provided by a source can be measured, we should find an appropriate way to do it.

Suppose we have two random variables which can take values from a finite alphabet each:

$$x_i \in \{x_1, x_2, ..., x_n\}$$
$$y_i \in \{y_1, y_2, ..., y_n\}$$

If both are statistically independent, the information about X provided by an event in Y is zero. On the other hand, if the occurrence of $Y = y_j$ determines completely the occurrence of $X = x_j$, then, the information the event Y provides about X is the same as the information provided by x_i.

Any measure of information we devise must fulfil the previous two conditions; the following function appears to be suitable:

$$I(x_i; y_j) = \log \frac{P(x_i|y_j)}{P(x_i)}, \text{ known as } \textit{mutual information} \tag{A.4}$$

If we take $\log_2$ the units of the mutual information are called bits.
Let us check if it really satisfies the two conditions previously exposed:

1. If there is independence between both events,

$$I(x_i; y_j) = \log \frac{P(x_i|y_j)}{P(x_i)} = \log \frac{P(x_i)}{P(x_i)} = 0$$

2. If occurrence of Y totally determines the occurrence of X, then

$$I(x_i; y_j) = \log \frac{P(x_i|y_j)}{P(x_i)} = \log \frac{1}{P(x_i)} = I(x_i), \text{ called } \textit{self-information} \tag{A.5}$$

We observe that a high-probability event conveys less information than other with lower probability.

It is also truth that the information about x_i provided by y_j is identical than the information provided by x_i about the occurrence of y_j.

$$I(x_i; y_j) = \log \frac{P(x_i|y_j)}{P(x_i)} = \log \frac{\frac{P(x_i,y_j)}{P(y_j)}}{P(x_i)} = \log \frac{P(y_j|x_i)}{P(y_j)} = I(y_j; x_i)$$

We can also define the *conditional self-information* as:

$$I(x_i|y_j) = \log \frac{1}{P(x_i|y_j)} \tag{A.6}$$

Therefore, the following relation holds true:

$$I(x_i; y_j) = I(x_i) - I(x_i|y_j)$$

A.4 Average mutual information and entropy

We can step further and define the average mutual information between X and Y as:

$$I(X;Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) I(x_i; y_j) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) \log \frac{P(x_i, y_j)}{P(x_i)P(y_j)} \tag{A.7}$$

It holds that $I(X;Y) \geq 0 \; \forall \; \{X, Y\}$.

Identically, we can obtain the *average self-information* as:

$$H(X) = \sum_{i=1}^{n} \log \frac{1}{P(x_i)} \tag{A.8}$$

Where X represents the alphabet of possible output letters from a source, $H(X)$ represents the average self-information per output and it is called the entropy of the source.

$H(X)$ is always less or equal than $log(n)$, where the equality holds when symbols are equally probable.

Finally, we can define the average conditional self-information (or conditional entropy) as:

$$H(X|Y) = \sum_{i=1}^{n}\sum_{j=1}^{m} P(x_i, y_j) \log \frac{1}{P(x_i|y_j)} \tag{A.9}$$

Again it is true: $I(X;Y) = H(X) - H(X|Y) \geq 0$.

The results previously exposed can be generalized to more than two variables as:

$$H(X_1X_2...X_k) = \sum_{i=1}^{k} H(X_i|X_1X_2...X_{i-1}) \tag{A.10}$$

That satisfies:

$$H(X_1X_2...X_k) \leq \sum_{m=1}^{k} H(X_m)$$

This information measures can easily be extended for continuous random variables by simply applying some little changes.

Appendix B

Matlab and TinyOS: getting them to work together

B.1 Introduction

In this chapter we will go through the different aspects which are needed to be taken into account when interconnecting Matlab and TinyOS, as well as the way in which they have to be used. We will show why Matlab is a suitable tool for interacting with WSNs.

Matlab is a scripting language in which large applications can be written and is interpreted, which means that it is slower than other languages like C or Java. However, being an interpreted scripting language is also part of what makes Matlab an appealing way to interact with a sensor network: the user can interact with the network by calling commands on the Matlab command line. In contrast, once a java application is started, it can only be controlled through a GUI.

It is possible to connect Matlab with the motes to receive and inject information from and to the network. The system architecture will consist of a sensor (or several) connected to the computer through the USB port, acting as a data-gathering node, and many others sensing nodes. The sink node will behave as base station receiving all the packets addressed to its network address. From Matlab we will be able to connect to this mote, listening to the desired packets in the network and filtering the rest of them.

We will also have full access to the message structure having the possibility of accessing/editing the packets'fields in a java-oriented syntax. It will be also possible to send packets to any sensor in the network, provoking reactions (if programmed beforehand) which will make our network reactive.

Last but not least, Matlab is a complete programming development environment having numerous tools for helping the developer in their work. Thus, Matlab appears as an ideal environment for an easy and fast development of a broad range applications.

B.2 Setting up the Matlab environment to use it with TinyOS and Java

Step 1 The Matlab directory structure provided by TinyOS was meant to mirror that of the tinyos-1.x directory. Each of the directories is meant to serve the following purposes:

APPS holds Matlab functions that were built for a certain tinyOS applications, e.g. oscilloscopeRF.

CONTRIB contains subdirectories to mirror tinyos-x.x/contrib for matlab applications.

LIB tools that correspond to tinyOS components in tos/lib.

TOOLS Matlab functions or apps that are generally useful but do not relate specifically to one app (e.g. 'listen.m').

UTIL functions (not apps) that may be shared among several other Matlab apps, eg. message processing utilities.

Add to your matlab directory in: *your_path_to_tinyos-1.x\tools\matlab* those directories above which are not present with the distribution.

Step 2 Create the file ***startup.m*** in the folder *your_path_to_Matlab\toolbox\local*. Edit the file with the following code:

```
flag=0;
global TOSDIR
TOSDIR='your_path_to_UCB\cygwin\opt\tinyos-1.x\tos';
addpath your_path_to_tinyos-1.x\tools\matlab;
addpath your_path_to_tinyos-1.x\tools\matlab\comm;
```

```
addpath your_path_to_tinyos-1.x\tools\matlab\apps;
addpath your_path_to_tinyos-1.x\tools\matlab\lib;
addpath your_path_to_tinyos-1.x\tools\matlab\util;
addpath your_path_to_tinyos-1.x\tools\matlab\tools;
addpath your_path_to_tinyos-1.x\tools\matlab\contrib;


defineTOSEnvironment;
```

Basically, what we do in this Matlab script is to set up the Matlab path and call the script ***defineTOSEnvironment.m***, which in turn, will initialize the Matlab comm (communications)stack. By giving the script the name ***startup.m*** and placing it in the aforementioned directory, we ensure the file is executed when Matlab starts up.

Step 3 Edit the file ***defineTOSEnvironment.m*** that you can find in *your_path_to_tinyos-1.x\tools\matlab* so that it looks like:

```
global DEBUG
DEBUG = 0;


global COMM
COMM.TOS_BCAST_ADDR = 65535;
COMM.TOS_UART_ADDR = 126;
COMM.GROUP_ID = hex2dec('7D');


defineComm;
```

Step 4 Copy the file ***comm.jar*** located in the following directory:
your_path_to_UCB\jdk1.4.1_02\j2sdk1.4.1_02\jre\lib\ext to
your_path_to_tinyos-1.x\tools\java

Step 5 Open the Matlab file ***classpath.txt*** and add the following pathes:

```
your_path_to_tinyos-1.x\tools\java
your_path_to_tinyos-1.x\tools\java\comm.jar
your_path_to_tinyos-1.x\tools\java\net\tinyos\message
```

Step 6 Copy the files ***win32com.dll*** and ***getenv.dll*** located in
your_path_to_UCB\jdk1.4.1_02\j2sdk1.4.1_02\jre\bin
and the folder *your_path_to_tinyos-1.x\tools\java\jni* in the Matlab folder
your_path_to_Matlab\sys\java\jre\win32\jre1.4.2\bin

Also copy the file ***javax.comm.properties*** located in the folder
your_path_to_UCB\jdk1.4.1_02\lib to the destination folder:
your_path_to_Matlab\sys\java\jre\win32\jre1.4.2\lib

Step 7 Edit the file ***cygwin.bat*** to include in the system classpath the following pathes:

your_path_to_Matlab\java\jar\jmi.jar
your_path_to_UCB\cygwin\opt\tinyos-1.x\tools\java\comm.jar
your_path_to_UCB\cygwin\opt\tinyos-1.x\tools\java

Step 8 The `net.tinyos.matlab.MatlabControl` class is needed to call Matlab commands from Java. Previously to compilation is mandatory to fix a bug in the ***MatlabControl.java*** file, by using the provided patch file.

Once it has been fixed, proceed to compile the folder *your_path_to_tinyos-1.x\tools\java\net\tinyos\matlab* by typing `make matlab`.

Please, note that you should have already included the jar file ***jmi.jar*** in your CLASSPATH environment variable (otherwise it returns the error: `package comm.mathworks.jmi does not exist`)

Step 9 Before you are able to use the Matlab functions ***(connect, receive, send and stopReceiving)*** provided along the TinyOS distribution, you will have to go through several typographical errors:

1. Edit the file ***receive.m*** as below:
   - Delete line 52
     (`moteIFs = [COM.sourceMoteIF{TF}];`) to place instead:

```
if isempty(TF)
    TF=0;
else
    TF=1;
end
if TF==0
    moteIFs=[];
else
    moteIFs = [COM.sourceMoteIF{TF}];
end
```

   - In line 63 take the transpose away.
2. Edit the script ***stopReceiving.m*** as follows:
   - Substitute the code lines:
     `COMM.globalFunction=COMM.globalFunction{~TF};`

```
COMM.globalMessageType=COMM.globalMessageType{~TF};
COMM.globalMessageName=COMM.globalMessageName{~TF};
```

by

```
COMM.globalFunction={COMM.globalFunction{~TF}};
COMM.globalMessageType={COMM.globalMessageType{~TF}};
COMM.globalMessageName={COMM.globalMessageName{~TF}};
```

- And the code lines:

```
for i=1:length(varagin)
    receive(functionName, message, varargin{i})
```

by

```
for i=1:length(varargin)
    stopReceiving(functionName, message, varargin{i})
```

Step 10 To finish with, it should be noticed that there still exist some bugs in the Java code used by the Matlab scripts that will make necessary to check the code carefully for every application.

B.3 Using the TinyOS java tool chain from Matlab

It is often easier to use an existing Java tool with TinyOS than to rewrite it in Matlab. Thus, we can use Matlab to launch the Serial Forwarder, the Oscilloscope application seen in the TinyOS tutorial [29], etc.

If we want to start the Oscilloscope application from Matlab, we should enter the following command:

```
net.tinyos.oscope.oscilloscope.main('125')
```

You should see the Java GUI open and connect to your serial forwarder.

When using Java from Matlab the syntax remains basically the same, except that there is no ”new” operator and functions with no arguments do not terminate with empty parenthesis ”( )”.

Every time a value is returned or passed to a Java method, a conversion of types takes place automatically (according to the information provided in

the Matlab help). Most of the arguments are passed by value except for the Java objects which are passed by reference.

Using Java classes from Matlab reveals two common bugs that should be fixed before you obtain a successful execution. While both of these bugs do not appear as such when running the java program from the default Java environment (i.e. starting a Java application from the command line), they do it when the program is called from Matlab. Thus, you will find them in many Java classes, including those in the TinyOS Java toolset.

A. Command Line Arguments In this section you will learn with a very simple example how to pass arguments from the Matlab command line to a Java method.

Imagine you want to run the Serial Forwarder to listen to packets arriving at port COM7. If you are calling the program from your default shell you should type the following command:

```
java net.tinyos.sf.SerialForwarder -comm serial@COM7:telos
```

To run the same program from the Matlab command line you should write the command:

```
net.tinyos.sf.SerialForwarder.main({'-comm','serial@COM7:telos'})
```

From the previous example we can deduce that the shell automatically packages up the command line arguments into string arrays before passing them to the main function of the class being called. In Matlab this has to be explicitly done by directly passing the arguments as string arrays.

In case we do not want to pass any argument, we should send a null array, which is done as:

```
net.tinyos.sf.SerialForwarder.main({ })
```

We get a null pointer exception! This is because the static main function in the main class of the SerialForwarder uses the string before checking if it is null. Since sending no command-line arguments from the shell does not result in a null string being passed, this normally does not cause an error. However, this should be fixed if this class were to be used from Matlab.

B. Virtual Machines You will have already realized that when we run a program in Matlab we do not use the command *java*. This is due to the fact that in Matlab we instantiate the objects within the same JVM (Java Virtual Machine) in which the Matlab session is running. This is only important for the `java.lang.System` class, which directly refers to the JVM you are running in; `java.lang.System.exit()` will kill your JVM, and therefore all the classes and your Matlab session! You will see this if you close the SerialForwarder window, because this causes a call to `System.exit()`. Hence, `System.exit()` should never be called.

B.4 Using Matlab with TinyOS

B.4.1 Preparing the message

Prior to connecting the computer to the network we need to build the messages to which we want to listen to. To illustrate this we will go through a simple example and we will make use of the available tools we have.

Let us imagine we have a network in which the base station is polling the nodes one by one and asking them to send back an application-specific data.

We need to construct two kinds of messages, one carrying the request for data to the sensors (*SimpleCmdMsg*) and another one carrying the data sent back by the sensors (*DataMsg*).

We can build two header files each one with the message structure we have devised, for example, if we list the code corresponding to the file ***SimpleCmdMsg.h***:

```
enum {
     AM_SIMPLECMDMSG = 18
};

typedef struct SimpleCmdMsg {
     uint16_t dst;
     uint16_t source;
     uint16_t seqno;
```

```
    uint8_t type;
    uint8_t focusaddr;
    uint8_t newrate;
    uint8_t hop_count;
    uint8_t bitsT;
    uint8_t bitsH;
    uint8_t bitsLtsr;
    uint8_t bitsLpar;
} SimpleCmdMsg;
```

Once we have the messages in the header files, we can use the MIG tool (see [29] for further information) to automatically generate Java classes which take care of the cumbersome details of packing and unpacking fields in the message's byte format. Using MIG saves you from parsing message formats in your Java application.

Once we have the output from MIG: *SimpleCmdMsg.java* and *DataMsg.java* we can proceed to compile them, obtaining the respective *.class* files.

Now we can easily instantiate these objects in Matlab.

B.4.2 Connecting Matlab to the network

This subsection does not intend to be a detailed guide of how to use Matlab with TinyOS because it would be a redundant work over that present in [29]. We will just try to provide a basic understanding of the overall working by continuing with the example we started in section B.4.1.

The first step is to connect your Matlab session to your network (namely to your base station). If you are working with Tmotes this can be done as:

```
connect('serial@COM7:telos');
```

Where, same as before, we are assuming that your base station is identified by the serial port `COM7` (you can use the command `motelist` in your cygwin environment to check what devices are connected to your computer).

Once you have done this, you can instantiate the MIG message, which is a Java class that is a subclass of `net.tinyos.message.Message`. In Matlab you can instantiate Java objects on the command line as follows:

```
dataMsg=net.tinyos.report.DataMsg
```

Now you are prepared to start receiving packets:

```
receive('handleMsg',dataMsg)
```

This command specifies the Matlab function ***handleMsg*** as the one in charge for handling the received messages, and the *DataMsg* messages as the ones to which the base station is listening to. Any other arriving packet will be discarded.

If we want to send a packet to a node:

```
send(3,simpleCmdMsg)
```

Where *simpleCmdMsg* is an instance of the class SimpleCmdMsg sent to the node with network address 3.

At this point, the *DataMsg* objects should be printed to your screen every time a message of this type is received. To stop this behaviour you can use the `stopReceiving` command to deregister your Matlab function as a message handler:

```
stopReceiving('handleMsg',dataMsg)
```

Finally, you can disconnect yourself from the sensor you are connected to with the command:

```
disconnect('serial@COM7:telos')
```

Appendix C

Acronyms

ADC	Analog to Digital Converter
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
CRC	Cyclic Redundancy Check
dB	decibel
DCO	Digitally Controlled Oscillator
DMS	Discrete Memory Source
DS-CDMA	Direct Sequence - Coded Division Multiple Access
IM	Initialization Module
IR	Infrared
IS	International System
Kbps	Kilobits per second
KTH	Kungliga Tekniska Högskolan
LMS	Least-Mean-Squares
LSB	Least Significative Bit
MAI	Multiple Access Interference
ME	Minimum Energy
MME	Modified Minimum Energy
MSE	Mean Square Error
OOK	On-Off Keying
PAR	Photosynthetically Active Radiation
PDA	Personal Digital Assistant

PN	Pseudorandom Noise
RF	Radio Frequency
SINR	Signal to Interference + Noise Ratio
SMA	Surface Mount Assembly
TSR	Total Solar Radiation
USB	Universal Serial Bus
VCP	Virtual Com Port
WSN	Wireless Sensor Networks

Bibliography

[1] C.H. Liu and H.H. Asada. "A Source Coding and Modulation Method for Power Saving and Interference Reduction in DS-CDMA Sensor Network Systems. In *Proc. of American Control Conf.*, volume 4, pages 3003–3008, May 2002.

[2] J. Kim and J.G. Andrews. "An Energy Efficient Source Coding and Modulation Scheme for Wireless Sensor Networks". In *IEEE* 6^{th} *Workshop on Signal Processing Advances in Wireless Communications*, pages 710–714, June 2005.

[3] Dragan Petrovic, Kannan Ramchandran, and Jim Chou. "A Distributed and Adaptive Signal Processing Approach to Reducing Energy Consumption in Sensor Networks". In *IEEE INFOCOM*, 2003.

[4] C. Toh. "Maximum Battery Life Routing to Support Ubiquitous Mobile Computing in Wireless AD-HOC Networks". In *IEEE Communications Magazine*, pages 138–147, June 2001.

[5] R. Shah and J. Rabaey. "Energy Aware Routing for Low Energy ad-hoc Sensor Networks". In *Proc. of IEEE WCNC*, March 2002.

[6] D.E.C. Intanagonwiwat and R. Govindan. "Directed Difussion: a Scalable and Robust Communication Paradigm for Sensor Networks". In *Proc. of IEEE MobiCom*, August 2002.

[7] G. Pottie and W. Kaiser. "Wireless Sensor Networks". In *Communications of the ACM*, 2000.

[8] M. Chu, H. Haussecker, and F. Zhao. "Scalable Information-driven Sensor Querying and Routing for ad hoc Heterogeneous Sensor Net-

works". In *IEEE Journal of High Performance Computing Applications*, 2002.

[9] R. Blahut. *Theory and Practice of Data Transmission Codes*. 1995.

[10] S. Toumpis and A.J. Goldsmith. "Capacity Regions for Wireless ad hoc Networks". In *IEEE Trans. on Wireless Communications*, volume 24, pages 3003–3008, May 2003.

[11] I.F. Akylidiz, W.Y. Sankarasubramaniam, and E. Cayirci. "A Survey on Sensor Networks". In *IEEE Communications Magazine*, pages 102–114, August 2002.

[12] E. Shih, H. Calhoun, S. Cho, and A.P. Chandrakasan. "Energy-Efficient Link Layer for Wireless Microsensor Networks". In *IEEE Computer Society Workshop on VLSI*, pages 16–21, April 2001.

[13] C. Erin and H.H. Asada. "Energy Optimal Codes for Wireless Communications". In *Proc. of the* 38^{th} *IEEE Conference of Decision and Control*, volume 5, pages 4446–4453, December 1999.

[14] Y. Prakash and S.K. Gupta. "Energy Efficient Source Coding and Modulation for Wireless Applications". In *Proc. or IEEE WCNC*, volume 1, pages 212–217, March 2003.

[15] M.B. Pursley. "Performance Evaluation for Phase-Coded Spread-Spectrum Multiple-Acces Communication - Part I: System Analysis". In *IEEE Transactions on communications*, volume COM. 25, pages 795–799, August 1977.

[16] S. Yao and E. Geraniotis. "Optimal Power Control Law for Multimedia Multirate CDMA Systems". In *IEEE* 46^{th} *Vehicular Technology Conference: 'Mobile Technology for the Human Race'*, volume 1, pages 392–396, April 1996.

[17] F. Santucci, G. Durastante, F. Graziosi, and C. Fischione. "Power Allocation and Control in Multimedia CDMA Wireless Systems". In *Telecommunication Systems*, volume 23, pages 69–94. Springer Netherlands, June 2003.

[18] C. Fischione, A. Bonivento, K.H. Johansson, and A. Sangiovanni-Vincentelli. "Cooperative Diversity with Disconnection Constraints and Sleep Discipline for Power Control in Wireless Sensor Networks". In *IEEE VTC*, 2006.

[19] A. Popoulis. *"Probability, Random Variables, and Stochastic Processes"*. McGraw-Hill, 1984.

[20] C. Fischione and M. Butusse. "Power and Rate Control Outage Based in CDMA Wireless Methods under MAI and Heterogeneous Traffic Sources". Technical report, Royal Institute of Technology (KTH), Stockholm, Sweden, 2006.

[21] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athen Scientific, 1997.

[22] M. Zuniga and B. Krishnamachari. "Analyzing the Transational Region in Low Power Wireless Links". In *IEEE SECON*, 2004.

[23] J.B. Andersen, T.S. Rappaport, and S. Yoshida. "Propagation Measurements and Models for Wireless Communications Channels". In *IEEE Communications Magazine*, pages 42–49, January 1995.

[24] Chipcon Products by Texas Instruments. *2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*, 2006.

[25] John G. Proakis. *Digital Communications*. Electrical Engineering. McGraw Hill International Editions, third edition, 1995.

[26] Moteiv Corporation. *Telos: Ultra low power IEEE 802.15.4 compliant wireless sensor module. Revision B : Humidity, Light, and Temperature sensors with USB*, 2004.

[27] S. Haykin. *Adaptive Filter Theory*. PrenticeHall, third edition, 1996.

[28] E. N. Gilbert. "Capacity of a Burst-Noise Channel". In *The Bell System Technical Journal*, volume 39, pages 1253–1265, September 1960.

[29] TinyOS Community Forum. *TinyOS tutorial*, September 2003.

[30] L. Schwiebert, S.K.S. Gupta, P.S.G. Auner, G. Abrams, R. Lezzi, and P. McAllister. "A Biomedical Smart Sensor for Visually Impaired". Technical report, IEEE Sensors, June 2002.

[31] B. Yang, S. Rhee, and H.H. Asada. "A Twenty-four Hour Tele-nursing System Using a Ring Sensor". In *Proc. of the IEEE International Conference on Robotics and Automation*, volume 1, 1998.

[32] B. Hashem and E. Sousa. "Performance Evaluation of DS-CDMA Systems Employing Adaptive Transmission Rate under Imperfect Power Control". In *Proc. of IEEE ICC'98*, 1998.

[33] K. Sipila et al. "Modelling the Impact of the Fast Power Control on the WCDMA Uplink". In *Proc. of IEEE VTC'99*, pages 1266–1270, May 1999.

[34] Holtzman. "A Simple, Accurate Method to Calculate Spread-Spectrum Multiple-Access Error Probabilities". In *IEEE Trans. Comm.*, volume 40, pages 461–464, 1992.