NTNU
Norwegian University of
Science and Technology

A Pragmatic Approach to Modulation Scaling Based Power Saving for Maximum Communication Path Lifetime in Wireless Sensor Networks

Raúl Malavia Marín

Master of Science in Electronics
Submission date: June 2008
Supervisor: Kjetil Svarstad, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Problem Description

For wireless sensor networks it is very important to find a system design for the node sensors that use as low power as possible. This is due to the fact that the nodes are utterly small and if they carry batteries at all, the capacity will be diminutive.

Tasks will be:

1.Research relevant literature
2.Suggest a node architecture for study
3.Suggest a simple network architecture for study
4.Develop a task representation suitable for computing power demands and constraints for nodes
5.Suggest and test one or a few alternative algorithms for leveraging module supply voltage with respect to scheduling and timing constraints.
6.Discuss the contribution with respect to earlier published solutions.

Assignment given: 15. January 2008
Supervisor: Kjetil Svarstad, IET

Preface

This last year spent in Norway has been fascinating in many senses. The Norwegian traditions and its nice people, the cold weather, the spectacular landscapes, and the life together people from many different countries, have been an unforgettable and rich experience.

I would like to express my special thanks to my supervisor Kjetil Svarstad for his continuous guidance and support along the way. Also, I would like to thank a lot my "Norwegian Family" for their support and the good moments that we have spent together. Furthermore, say that without my "Valencian Family" it would have been impossible to overcome all the challenges and problems presented along these five past years.

Last but not least, I want to thank to my parents and my sister for their, and I cannot stress this enough, endless support during the whole career.

Abstract

The interest in Wireless Sensor Networks is rapidly increasing due to their interesting advantages related to cost, coverage and network deployment. They are present in civil applications and in most scenarios depend upon the batteries which are the exclusive power source for the tiny sensor nodes. The energy consumption is an important issue for research, and many interesting projects have been developed in several areas. They focus on topology topics, Medium Access Control or physical issues. Many projects aim at the physical layer where the node's power consumption is optimized through scaling the modulation scheme used in node communications. Results show that an optimal modulation scheme can lead to the minimum power consumption over the whole wireless sensor network. A usual simplification in research is to target individual paths and not take into account the whole network. However nodes may be part of several paths, and therefore nodes closer to the sinks may consume higher amounts of energy. This fact is the chief motivation of our research, where modulation scaling over the nodes with more energy is performed in order to increase the lifetime of the nodes having lower energy reserves. Simulation results showed typical values of path lifetime expectancy of 50 to 120 percent higher than comparable power-aware methods.

Contents

1 Introduction **9**

1.1 About This Thesis . . . 9

1.2 Structure and Goals of This Thesis . . . 9

2 Background and Related Work **11**

2.1 Wireless Sensor Networks . . . 11

2.1.1 Main Aspects . . . 11

2.1.2 Topology . . . 12

2.1.3 Wireless Channel Characteristics . . . 14

2.1.4 Sleep Scheduling . . . 17

2.1.5 Data Processing . . . 17

2.2 Previous Related Work . . . 18

2.2.1 High layer Techniques . . . 18

2.2.2 MAC layer . . . 21

2.2.3 Physical Layer . . . 23

3 Problem Definition **31**

3.1 Scenario . . . 31

3.2 Energy Model . . . 33

3.2.1 Modulation in Wireless Sensor Networks . . . 33

3.2.2 The Energy Model . . . 36

3.2.3 Energy model for odd values of k . . . 38

3.3 Our Approach . . . 38

4 Proposed Solution **43**

4.1 Structure . . . 43

4.2 Description of Sub-files . . . 44

4.2.1 Fixedpoint.m . . . 44

4.2.2 Energy_model.m . . . 44

4.2.3 Energyconstraints.m . . . 44

4.2.4 Optimal_energy.m . . . 45

4.2.5 Optimal_b.m . . . 45

4.2.6 Balance_b.m . . . 46

4.3 The main file . . . 47
4.3.1 Node Structure . . . 48
4.3.2 General Description . . . 49
4.3.3 Details . . . 50

5 Performance Analysis **55**
5.1 Network Simulator . . . 55
5.2 Theoretical Path . . . 57
5.3 Tree-based Topology . . . 62
5.3.1 Data Aggregation Tree . . . 62
5.3.2 Random tree . . . 67
5.3.3 Conclusions for Tree Topology . . . 71
5.4 Multihop Networks . . . 72
5.4.1 Exponential Distribution 1 . . . 73
5.4.2 Exponential Distribution 2 . . . 78
5.4.3 Conclusions for exponential distributions . . . 81

6 CONCLUSIONS **83**
6.1 Main Findings and Results . . . 83
6.2 Future Work . . . 85

A Matlab Code for the Algorithm **89**
A.1 Main.m . . . 89
A.2 Energyconstraints.m . . . 100
A.3 Optimal_energy.m . . . 101
A.4 Optimal_b.m . . . 103
A.5 Balance_b.m . . . 106
A.6 Energy_model.m . . . 110

B Matlab Code for Simulator **111**
B.1 Theoretical Path Simulator . . . 112
B.2 Tree and Exponential Distribution Simulator . . . 115

List of Figures

2.1 Tree-based Topology 12
2.2 Different topologies 14
2.3 Realistic packet reception rate statistics with respect to inter-node distance. Reprinted from [3]. 15
2.4 A realistic node with non regular or isotropic packet reception. Reprinted from [5]. . . . 15
2.5 Simple model of a radio 16
2.6 Example of virtual grid in GAF 18
2.7 State transitions of STEM for a particular node 19
2.8 Radio set up for a sensor node 19
2.9 The LEACH cluster-based routing technique 21
2.10 Periodic listen and sleep 22
2.11 Time division for scheduling in FPS 23
2.12 Energy-delay trade-off in DMS 24
2.13 Before and after applying scaling factors 25
2.14 Energy performance. Reprinted from [14]. 26
2.15 Relationship between thresholds and modulation. Reprinted from [15]. 27
2.16 Energy function for transmitting one packet. Reprinted from [16] 28
2.17 Pseudo code. Reprinted from [17] 30
2.18 Energy functions of the nodes of a path and the optimal solution. Reprinted from [17] . 30

3.1 Multi-hop wireless sensor network 32
3.2 Prob. of symbol error vs. SNR for M-PSK and M-QAM. Reprinted from [18]. 33
3.3 Prob. of bit error vs. SNR for M-FSK. Reprinted from [18]. 34
3.4 Transmit power vs. bandwidth efficiency in fading channel. Reprinted from [19]. . . . 35
3.5 Energy curves for long-range communications 40
3.6 Energy curves for short-range communications 40
3.7 Relationship between the overspending in the node with more remaining energy and the saving in node with less energy 41
3.8 Discrete points (E_i, τ_i)in an energy curve 42

4.1 Flow diagram of optimal_b.m 46
4.2 Flow diagram of balance_b.m 47
4.3 Flow fiagram of main.m 49

5.1 Simple path of nodes 57
5.2 Behaviour of our method 58
5.3 Energy curves 60
5.4 Behaviour for dynamic constraints 61
5.5 Spanning tree 62
5.6 Performance for different latency constraints 66
5.7 Random tree topology 67
5.8 Performance for different latency constraints 71
5.9 Exponential consumption along the path of nodes 73
5.10 Performance for several latencies 76
5.11 Consumption distribution along the path 78
5.12 Performance for several latencies 80

6.1 Performance summary 84
6.2 Energy curves 84

A.1 Flow diagram of main.m 89
A.2 Flow diagram of optimal_b.m 103
A.3 Flow diagram of balance_b.m 106

Nomenclature

BER	Bit Error Rate
CMOS	Complementary Metal Oxide Semiconductor
DCS	Dynamic code scaling
DMS	Dynamic Modulation Scaling
DVS	Dynamic Voltage Scaling
EDF	Earliest Deadline First
FLA	Frame Length Adaptation
FPS	Flexible Power Scheduling
FSK	Frequency Shift keying
GAF	Geographic Adaptative Fidelity
LEACH	Low-Energy Adaptive Clustering Hierarchy
OFDM	Orthogonal Frequency Division Multiplexing
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QoS	Quality of Service
S-MAC	Sensor Medium Access Control protocol
SNR	Signal to Noise Ratio
STEM	Sparse Topology and Energy Management
TDMA	Time division multiple access

Chapter 1

Introduction

1.1 About This Thesis

Our work is focused on Wireless Sensor Networks. We deal with low power techniques in order to achieve a good management of the energy resources and therefore to attain a good performance regarding life-time of the whole sensor network.

Many recent approaches have centred on the physical layer. Design of protocols in this layer implies not having awareness of the whole network. Thus, the existing approaches do not take into account the fact that a node inside a network may be a node in several paths and thus energy consumption is dependent of how many paths are crossing it. Hence, power consumption among paths leads to have different energy consumption in different nodes.

Our research aims at the physical layer where the above issue is taken into account, and where the modulation scheme of the nodes is scaled in order to achieve a good power consumption management. Emphasis is put on targeting the nodes with higher consumption in order to decrease their modulation parameters while rising up low-consumption nodes' modulation.

1.2 Structure and Goals of This Thesis

Several tasks were entrusted us at the beginning of this Master Thesis:

1. Research relevant literature.
2. Suggest a node architecture for study.
3. Suggest a simple network architecture for study.
4. Develop a task representation suitable for computing power demands and constraints for nodes.
5. Suggest and test one or a few alternative algorithms for leveraging module supply voltage with respect to scheduling and timing constraints.
6. Discuss the contribution with respect to earlier published solutions.

Task one was performed carefully in order to get a wide notion of which problems are present in Wireless Sensor Networks. Then, tasks 2, 3 and 4 were implemented in Matlab. At first we though about using network simulators to analyse our algorithm, but due to the amount of time required to understand and use such simulators, and due to no prior experience with them, we decided to develop our own simulator and network node structure. The main network architectures proposed were the Tree-based topology and the Many-to-one pattern topology. Task 5 was satisfied suggesting an algorithm that trades off latency against path lifetime, based on scaling the modulation scheme of the nodes. Finally, task 6 was carried out testing different protocols and comparing their performance to ours.

This thesis is structured as follows: Firstly, we give a thorough overview of the background concepts, as well as familiarizing the reader with some of the related research in the WSN area. Secondly, the mathematical model which the analysis is based on is described. The model is an energy-latency relationship for an M-QAM modulation, where both even and odd modulation parameters are considered. A demonstration is carried out in order to show that the approximation done for odd modulation values fits accurately into our model. Thirdly, we define the problem and we propose a solution. The latency constraints must be satisfied while at the same time meeting the energy constraints. Finally, a performance analysis is carried out in Matlab in order to confirm either the benefits or drawbacks of our approach with respect to previous research in the area. Comparisons regarding earlier published solutions are performed. Discussions and explanations are provided for all the results.

Chapter 2

Background and Related Work

In this chapter we give a short description of necessary concepts and related work in order to be able to understand the analysis and work that is to follow in the remaining of this thesis.

2.1 Wireless Sensor Networks

In recent years, interest in WSNs has increased drastically due to their advantages of the cost, coverage and network deployment. The development was originally motivated by military applications where the rapidly deployment, self-organization and fault tolerance of nodes provided an efficient way for battlefield surveillance and target tracking, detection of biological and nuclear weapons, and other military operations. However, as many technologies born in military, WSNs were extended to civil domains. For instance, ecological habitat monitoring, structural and seismic monitoring, or industrial networked sensing are some examples of current applications working with WSNs.

2.1.1 Main Aspects

WSNs are interesting from an engineer perspective, because they present serious key design challenges. The most important aspects related to our work are described in the following:

Lifetime

Sensor nodes are battery driven and hence operate with a limited energy resource. In large-dense sensor networks it is infeasible to replace batteries when a sensor is down. In practice, it will be necessary in many applications to provide guarantees that a network with unattended wireless sensors can remain operational without any replacements for several years. For instance, in wild and unreachable areas, such the Antarctica or the deepest zones of the Atlantic Ocean, sensors can be easily deployed in order to form a large-dense sensor network and sense seismic waves, temperature or other parameters as well. In these scenarios, the replacement of the battery of a sensor node would be highly expensive. Thus, many protocols have been developed as described in section 2.2 in order to have longer lifetime.

Deployment

The network must be deployed keeping in mind two main objectives: coverage and connectivity [1]. Both describe the robustness of having always a path between every pair of nodes. They can be easily controlled if the deployment is carried out via careful hand placement of network nodes. This is what we call structured deployment approach. On the other hand, randomized deployment approach can be used for large scale applications where, for instance, nodes can be dropped from an aircraft. Another aspect to be mentioned is the heterogeneity or homogeneity of the deployment. Not always all the nodes offer same performance due to the characteristics of the network topology. For example, in a single-hop star topology, the sink has a higher traffic load than the sensing ones. Therefore, the sink node must have better performance characteristics in order to cope with the requirements.

Self-configuration

Nodes in a wireless sensor network have to be able to configure their own network topology; localize, synchronize, and calibrate themselves; coordinate inter-node communication; and determine other important operating parameters [1]. Also they must be able to adapt themselves to the environmental conditions and unexpected situations in order to keep the performance negotiated and have a robust network. After deployment it is common in wireless sensor networks having topology changes due to changes in sensor nodes position, reachability, available energy, device failure or energy depletion [2]. Thus, many protocols provide auto-management techniques for avoiding collisions, adapt the radio-coverage of sensors, replace dropped nodes, and many other aspects. Some examples are described in section 2.2.

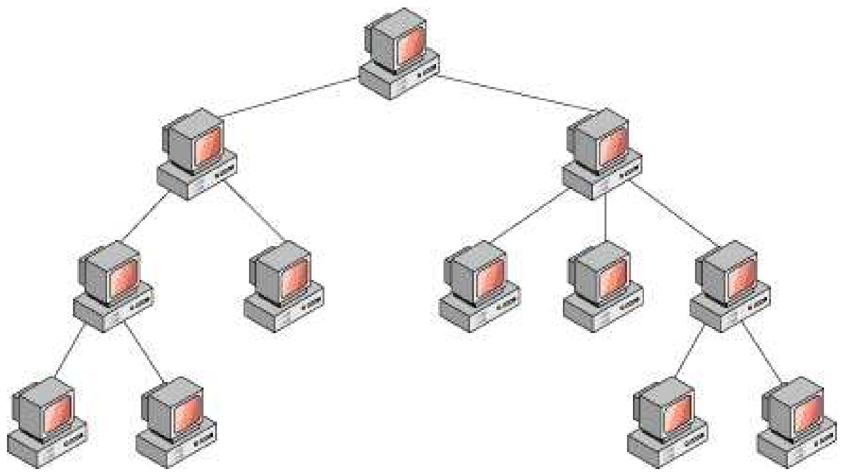

Figure 2.1: Tree-based Topology

2.1.2 Topology

For larger areas and networks, multi-hop routing is necessary. It provides robustness and scalability properties. Also, mesh networks are used: they are regularly distributed networks that generally allow transmission only to a node's nearest neighbour. Therefore several topologies are usually used in WSN projects. Some of them are described in the following:

Single-hop Star Topology

It is the simplest WSN topology (Fig. 2.2a). It follows the common pattern "many to one" that is given in many WSN configurations. Every node in this topology communicates its measurements directly to the gateway. The limitations of this topology are its poor scalability and its weak robustness properties. For instance, in larger areas, nodes that are distant from the gateway will have poor-quality wireless links. However, this issue can be solved if we use hierarchy as in tree-based topologies.

Tree-based Topology

It can be considered as a hierarchy of several single-hop star topologies (Fig. 2.1). Therefore, a node in this configuration has the duty of receiving and forwarding packets from its lower level interconnected star topology. Higher level nodes must support heavier traffic loads. In WSN it means that energy consumption will be drained faster in higher levels than in lower ones.

Multi-hop Mesh-based Topology (arbitrary/grid)

Nodes in these networks are generally identical. This topology brings multiple routing paths between nodes, what means that these nets are robust to failure of individual nodes or links. An additional advantage is that certain nodes can be designated as "main nodes" that take on additional functions. If a main node is disabled, another node can then take over its duties. Sensor nodes could be either thrown in mass or placed one by one. If they are placed in a random way, nodes form an arbitrary mesh graph (Fig. 2.2b), provide a better flexibility of arrangement and promote self-organization and fault tolerance. If not, they can form a more structured communication graph such as the 2D grid structure shown in Fig. 2.2c.

Clustered-based Topology

One of the most convincing architectures for WSN is a deployment architecture where multiple nodes within each local regions report to different cluster-heads (Fig. 2.2d). This approach needs the heterogeneity of the sensors in order to present cluster-heads nodes with higher performance. As in star or tree topologies, as we get close to the sink node in each hierarchy, the consumption of energy increases. Several algorithms have been developed in order to solve this issue (see section 2.2). The main advantage of this topology is that it decomposes a large network into separate zones. Within each cluster there could be either single-hop or multi-hop communication. Once data reach a cluster-head they would then be routed through the second-hierarchy network formed by several cluster-heads. There can be as many hierarchies as desired, depending on the size of the network.

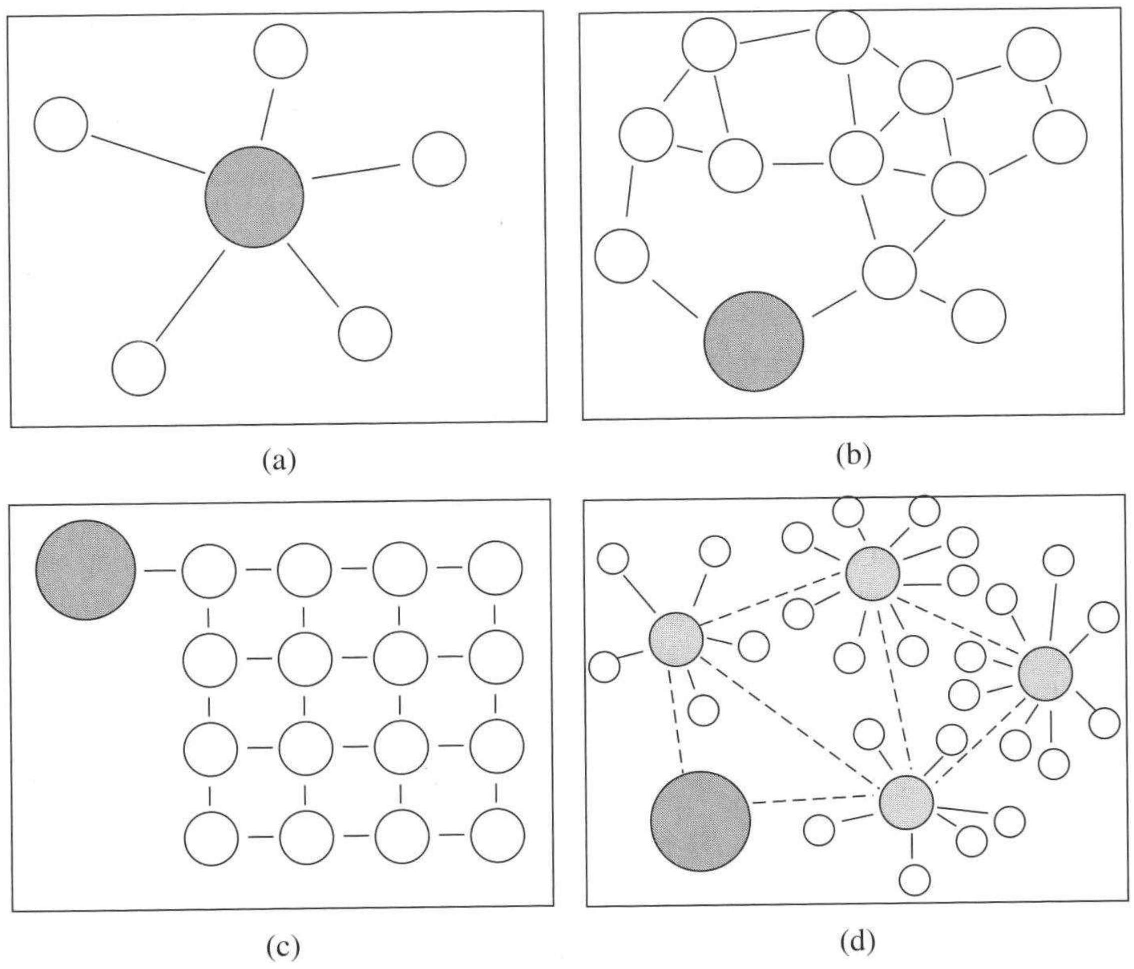

Figure 2.2: Different topologies

2.1.3 Wireless Channel Characteristics

Wireless communication imposes important challenges due to the expensive, variable and harshly link conditions. In order to be able to understand wireless sensor network protocols we focus on three important topics:

Link Quality

Basically two nodes have a perfect link[1] if they are within communication range R and a non-existent link if they are outside this range. However it is not such simple. Empirical observations [3, 4, 5] show that the quality of a given link is quite sensitive to many factors such as node positions (see Fig. 2.3), surrounding environment, individual hardware differences, asymmetrical links[2] (see Fig. 2.4) or changes over time.

[1]We define a perfect link as a receiver that has 100% packet reception rate.

[2]An asymmetrical link presents a non regular or isotropic packet reception. This means that a node can have a high link quality in some directions and low link quality in other ones.

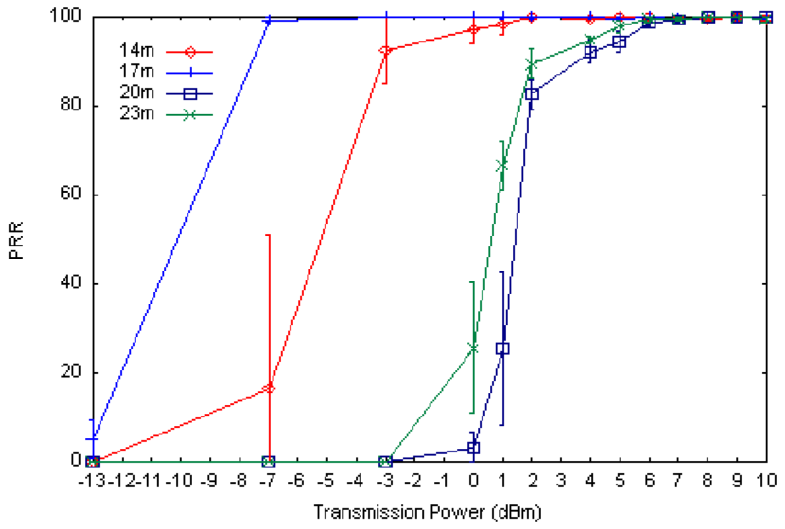

Figure 2.3: Realistic packet reception rate statistics with respect to inter-node distance. Reprinted from [3].

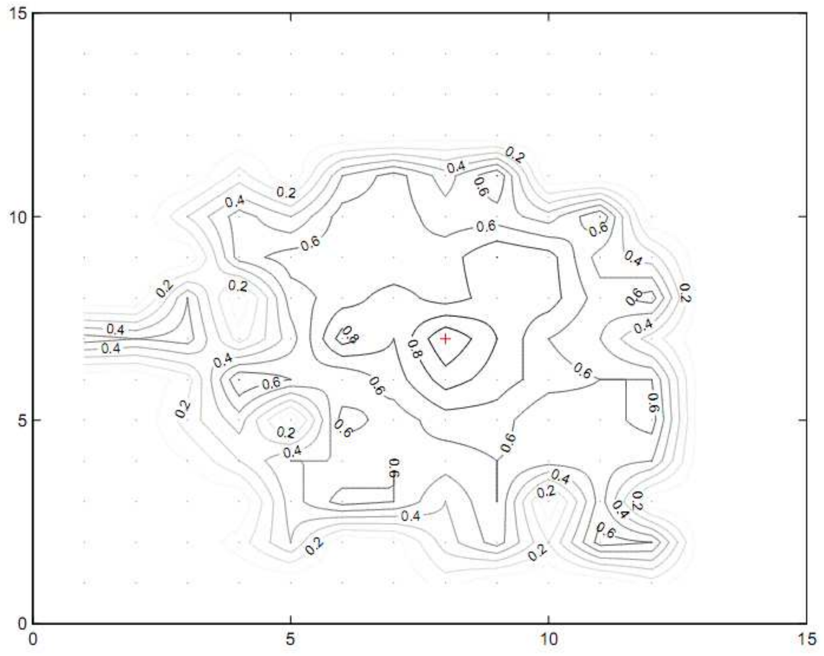

Figure 2.4: A realistic node with non regular or isotropic packet reception. Reprinted from [5].

Energy

It is important to know what the sources of power consumption in WSN are. The simplest radio model (see Fig. 2.5) considers that the energy consumed by the radio of a node has two components:

- Electronic power (P_E): the energy consumed by digital and analog components that carry out the RF, baseband and protocol processing.
- Transmit radio power (P_{RF}): the energy consumed by the radio power amplifier in the transmit path.

The energy associated with transmission or reception of a bit can be expressed as in equations shown below

$$E_{bit(tx)} = (P_{RF} + P_{E(tx)}) * T_{bit} \tag{2.1}$$

$$E_{bit(rx)} = P_{E(rx)} * T_{bit} \tag{2.2}$$

The required irradiated power (P_{RF}) depends on several factors, such as data rate, bit error rate, noise, interference, modulation scheme, receiver architecture and the path loss suffered by the signal. Last factor determines how the signal decays with the receiver-transmitter separation d. It is well known [6] that this decreasing of signal power follows a $1/d^{\gamma}$ fashion, where γ is the path loss exponent ($=$ 2 in free space and $\approx$ 2-4 in real life channels). Therefore the signal power at a certain distance is proportional to r^{γ} what means:

- It is more efficient to perform local communications (multihop networks).
- For wireless communications over large range, the communication energy will be dominated by the RF term P_{RF}.
- For short range wireless communications, the electronic power terms P_E would dominate the communication energy.

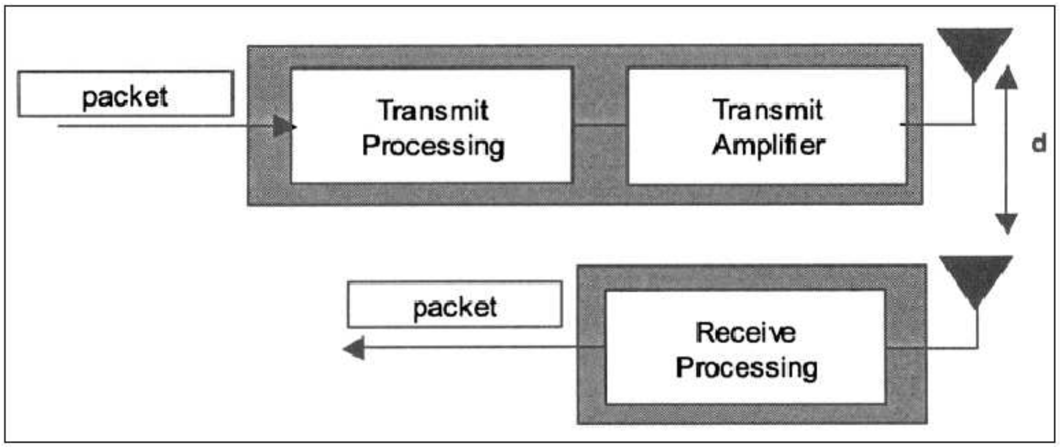

Figure 2.5: Simple model of a radio

Interference

In wireless sensor networks, a node's packet transmission is considered as a transmission over a circular ideal range. Sending data to a specific node implies to be interacting with other nodes that are in the nearby area. Hence, interferences between different sending nodes can appear. In practice, radios are capable of receiving packets without interference error, even when other packets are transmitted by neighbours.

2.1.4 Sleep Scheduling

An essential characteristic of wireless communications is that they provide a shared medium. The management of the channel resources in order to ensure an efficient utilization of the shared bandwidth is performed through MAC protocols. Standard MAC protocols are based on throughput, delay, fairness and reliability as objective functions. They are not able to recognize that, in a WSN, sensors are not competing for taking the channel, but cooperating. Hence, low-power wireless communications need power-aware MAC protocols that take care with emphasis energy efficiency.

WSN MAC protocols rely on the intuitive strategy of shutting down the radio when it is not used in order to save energy. They directly control the operational mode in which the node is at any instant of time. Transmit, receive, idle listening and sleeping are the available possibilities. Proper management of these states can be observed in protocols described in section 2.2.

On the other hand, from the point of view of network connectivity and coverage parameters, redundant nodes can be shut down while keeping the same performance of the network and achieving an efficient management of energy resources. Several protocols concerning network layer deal with this topic. In later sections more information is offered.

2.1.5 Data Processing

In wireless sensor networks, although the limited processing characteristics of sensors, they often posses enough intelligence to perform some rudimentary processing of data as it is received, in order to provide a more effective and efficient sensing. There are many motivations for the use of data processing in WSN. For instance, when multiple sensors are gathering the same phenomena sometimes the information sensed has a high level of correlation. Therefore, if data is processed as it is passed through the network, it can be compressed thus reducing the communication load. This compression process is described as aggregation. An example is a *data aggregation tree* where data is aggregated in those nodes that receive information from different sources (leaves).

Furthermore, data processing must provide to a sensor node the possibility of processing each others partial data and allow comparing with its local observations, in order to bring the possibility of taking decisions as dropping incoming data or auto-disconnecting itself. For instance, a large-dense wireless sensor network that presents lots of redundant nodes and should be shut down. Another example is a sensor network that is designed to detect brush fires. It has only to sense if a fire has started. Once a fire is detected, this information is forwarded to the user quickly.

2.2 Previous Related Work

From not so many years, when wireless technologies and VLSI design became feasible, wireless sensor networks have undergone an explosive growth in interest in both academia and industry. Many efforts have been done on the subject. In order to familiarize the reader, an overview of several power-aware mechanisms is described. A classification into high layer techniques (routing, transport and application layers), MAC and physical layer ones is performed. A deep description in the physical layer techniques will be carried out as it is the layer of our interest for this Master Thesis.

2.2.1 High layer Techniques

One characteristic of high layer techniques is that they consider the network as a whole and consider end-to-end communications through the network instead of individual nodes or links. They can be classified as energy-aware topology management techniques or energy-aware routing techniques. In the following, some relevant methods are described.

Topology approaches

The idea is that not all the nodes in a network need to have their radios active all the time in order to perform multihop packet forwarding. The goal is to coordinate the sleep transitions of all the nodes, while ensuring that data can be forwarded efficiently when desired. An example of this is given in dense-wireless sensor networks, where redundant nodes can be shut down while keeping the capacity of the network. Some mechanisms based in the topology of the network are described in the following:

GAF (Geographic Adaptative Fidelity) [7]
It relies on the fact that nearby nodes in a dense network can be equivalent and therefore redundant ones can be shut down. Each node uses location information to associate itself with a "virtual grid" (see Fig. 2.6), where all nodes in a particular grid square are equivalent with respect to forwarding packets. Nodes in the same grid then coordinate with each other to determine who will sleep and how long. The goal of GAF is therefore to ensure that a single node with the highest remaining lifetime is awake in each virtual grid. The features of GAF present a trade-off between node-density and lifetime. The higher the node-density is the higher lifetime can be achieved.

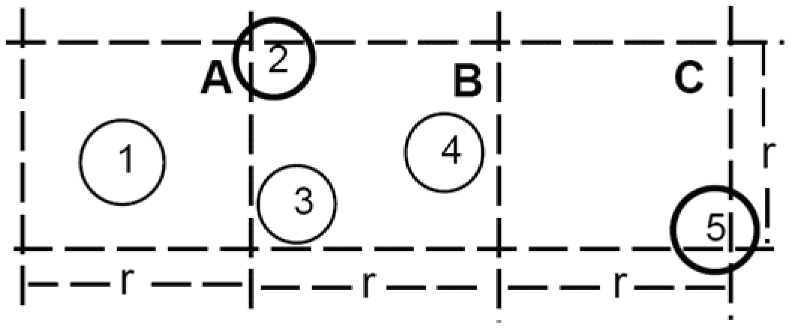

Figure 2.6: Example of virtual grid in GAF

STEM (Sparse Topology and Energy Management) [8]

It exploits the fact that most of the time, the network is only sensing its environment waiting for an event to happen. Therefore it would be ideal to have only working the sensing unit and some pre-processing circuitry of the node until a possible event would be detected. Hence, STEM relies on keeping the nodes' radio off until the processor decides that the acquired information needs to be forwarded. The main problem with this policy comes when a node wants to transmit data to other node which radio is shut down because it has not sensed the same event. The solution proposed by STEM is to turn on each node periodically for a short time in order to listen if someone wants to forward data. In the Fig. 2.7 is presented an example of how the wake up is carried out:

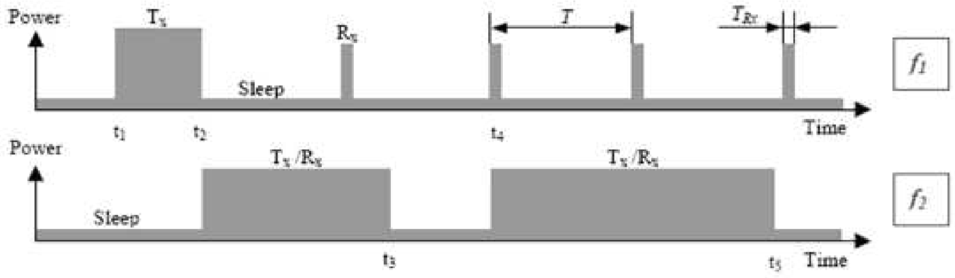

Figure 2.7: State transitions of STEM for a particular node

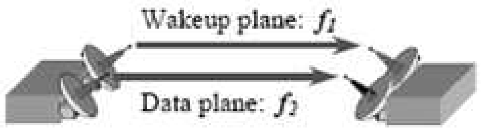

Figure 2.8: Radio set up for a sensor node

Two different frequencies are used (see Fig. 2.8), one for performing the wake up process, and another for transmitting data. The "initiator node[3]" sends out beacons in band *f1* to the "target node[4]". Meanwhile, the target's radio in band *f1* wakes up periodically from its sleep state. When the beacon is received by target just in one of these periodical checks, it responds to the initiator and the transmission starts (t2) in band *f2*. In t4 the inverse process is shown. The same node that was sending information before, has detected that other neighbour wants to transmit data to him and therefore the reception is performed in band *f2*.

[3]The node that wants to wake up one of his neighbours for transmitting data.

[4]The node that initiator node wants to wake up

The wake up phase shows that a set-up latency is added to the transmission time of the information. Hence, STEM presents a trade-off between energy savings and set-up latency.

Routing Approaches

Traditional routing was designed in order to satisfy metrics like minimum hop, shortest delay, maximum link quality, etc. However, wireless sensor networks need to achieve other objectives such as minimum energy consumption per node or extended lifetime. Some studies are shown in order to give to the reader a general overview of how power-aware metrics can be attained:

Multihop Routing

The free space loss models of RF signal show that the radiated power is proportional to the square of required transmission distance. Hence, in many WSN scenarios, it is more power efficient to emit low strength RF signals to travel a short distance and be relayed a number of times than transmitting high strength signals for a long range.

Lifetime-maximizing Routing

Many researches have been focused on minimizing the total consumed energy to reach the destination, which minimizes the total energy used per packet or flow transmitted. However, they do not realise that if all the traffic is routed through the optimal path or paths, their nodes will drain energy quickly while other nodes, perhaps more power hungry, will remain intact. Chang and Tassiulas [9] propose a technique called Flow Augmentation (FA) algorithm that takes into account the minimum total consumed energy and the node's remaining battery level. They introduce the following link metric

$$C_{i,j} = W_{i,j}^{x} * R_i^{y} * E_i^{z} \tag{2.3}$$

Where W is the energy expenditure for unit flow transmission over the link, R is the residual energy (remaining battery level) at the transmitting node i, and E is the initial energy of the node i.

This link metric captures a wide range of metrics (see 2.1). The path cost is computed by the summation of the link costs on the path, and the algorithm is implemented with a shortest path method that includes the distributed Bellman-Ford algorithm. Simulations results suggest that non-zero x and relatively large $y = z$ terms provide the best performance (e.g. $(1, 50, 50)$).

FA(x,y,z)	Meaning
FA(0,0,0)	Minimum hop path
FA(1,0,0)	Minimum transmitted energy path
FA(·,x,x)	Normalized residual energy is used
FA(·,·,0)	Absolute residual energy is used

Table 2.1: The meanings of the parameters of the algorithm FA

LEACH (Low-Energy Adaptive Clustering Hierarchy) [10]

It is a mechanism based in clustered-topology. The cluster-heads (see Fig. 2.9) periodically collect and aggregate the data from nodes within the cluster using TDMA, before sending them to the sink (directly or through multiple hops). It is easy to see that the unlucky sensors chosen to be cluster-heads would die quickly, ending the useful lifetime of all nodes belonging to those clusters. Thus LEACH provides a randomized periodically rotation of cluster-heads in order to distribute the duty of a cluster-head among all the nodes of the same cluster. The benefits are that the battery of a single sensor is not drained and the life-time of the network is increased.

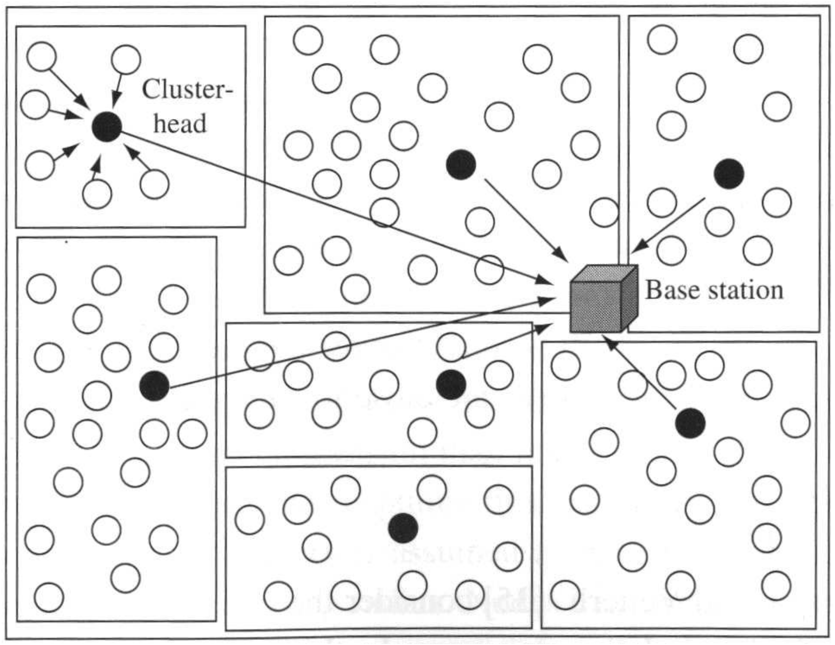

Figure 2.9: The LEACH cluster-based routing technique

2.2.2 MAC layer

Traditional MAC protocols manage the radio interface in order to achieve an efficient utilization of the shared bandwidth. However, WSN MAC protocols also need to ensure energy conservation. There are several sources of energy waste in MAC layer. The first one is collision. When a transmitted packet is corrupted it has to be discarded, and the retransmissions increase energy consumption. The second source is overhearing, meaning that a node picks up packets that are destined to other nodes. The third source is control packet overhead. The last major source of inefficiency is idle listening, i.e., listening to receive possible traffic that is not sent.

MAC protocols share the channel in one of two ways. In the first category are the protocols that rely in random channel access. A main feature is that the receiver is always listening if someone wants to send him data. In the second category are protocols based on time division multiplexing access (TDMA) where nodes see time as slots. An N-periodic schedule (fixed or dynamic) is used in order to allow nodes to take the use of the channel. TDMA is energy efficient because it is known when the radio will be idle and therefore when to turn it off.

MAC protocols directly control which mode the radio of a node is running. Thus, the node's radio can be in either transmit, receive, idle listening, and sleeping states. At the communication distances typical in WSN, idle listening has a similar cost to transmit and receive states. Hence, shutting down the radio when not used is an obvious strategy for saving energy.

S-MAC (Sensor Medium Access Control protocol) [11]

It is a wireless MAC protocol designed specifically for WSN. It tries to avoid problems like collisions, overhearing or idle listening states. It is based in employing a periodic cycle of sleep-listen states as in Fig. 2.10. During sleep state, radio is turned off and a timer is set to awake it later. In S-MAC, nodes exchange their schedules with the immediate neighbouring nodes in order to ensure that all neighbours can talk to each other even if they have different schedules. The methodology is the following one: a node waits for receiving a schedule of other nodes, if not, it chooses its own schedule and broadcasts before going to sleep. If a node receives a schedule before choosing its own one, it adopts that schedule and rebroadcasts before sleeping. If it receives it after choosing its own schedule, then it adopts both ones but only broadcasts its own schedule. Due to the sleep schedule of nodes an increase in latency appears, having a trade-off between energy and latency.

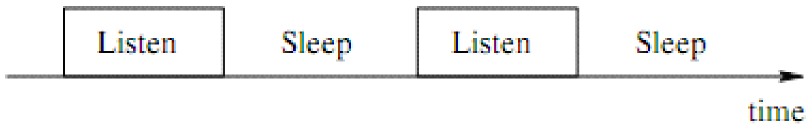

Figure 2.10: Periodic listen and sleep

FPS (Flexible Power Scheduling) [12]

It exploits a tree-based topology that enables a distributed algorithm using no centralized control, has adaptative schedules, and implements two-layer architecture. It uses fine grain medium access control in order to handle the channel and coarse grain scheduling at the routing layer to schedule the radio on-and-off during the idle times. In FPS, time is broken into cycles and slots (see Fig. 2.11). Each node maintains a local power schedule of what operations (receive, transmit, and idle) it performs over the course of a cycle. The schedule of a node is based on the knowledge of how many slots needs for receive data from its sons (demand), and how many slots are required for transmit its data plus the data of its sons (supply). Once demand and supply are known, a schedule of each cycle is performed, and in idle states the radio is turned off. More details are provided in the bibliography [12].

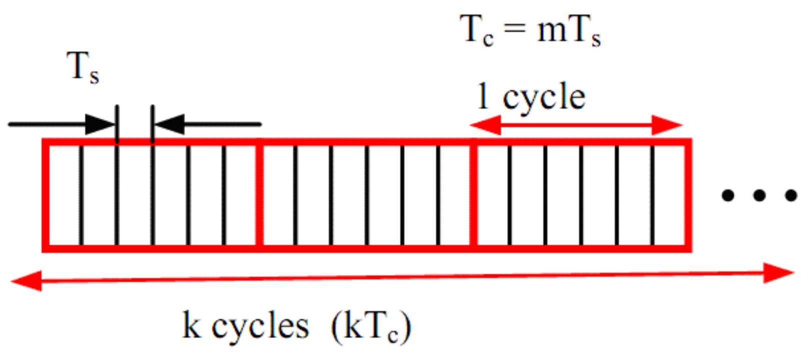

Figure 2.11: Time division for scheduling in FPS

2.2.3 Physical Layer

There exists a powerful class of dynamic power management techniques for radios that exploit control knobs providing similar energy-speed relationships as in CMOS circuits. In other words, use lower rates expending more time is more energy-efficient than use higher rates in less time. Examples of these control knobs are Dynamic code scaling (DCS), Frame Length Adaptation (FLA) or Dynamic Modulation Scaling (DMS). So, in DCS the forward error correcting code is scaled in order to use a lower-rate code. It results in lower energy consumed but with higher transmission time. In FLA, it is considered that operating with a fixed frame length is inefficient due to the fact that it is better to adapt the length to the channel conditions. So for example, smaller frames have better chance of successful transmission and depending on the channel conditions specific lengths are computed in order to maximize the goodput . Finally, DMS is another technique that is based in the modulation used for transmitting data. All the work of this thesis is based in modulation scaling. Consequently a wide description of it and many examples of use must be explained. In the following, the origin of DMS is explained, then the relationship between energy and time is graphically shown and finally several examples are described.

Dynamic Modulation Scaling

Scaling the supply voltage, Dynamic Voltage Scaling (DVS), is the most common circuit technique to offer both low energy consumption and energy awareness in embedded processors or real-time operating systems. It is based on schedule voltage while maintaining time and throughput constraints in order to achieve lower power consumption. The previous methodology can be extrapolated to communications systems. In Dynamic Modulation Scaling, the modulation parameter plays the role of the voltage in DVS and is scheduled in order to achieve lower energy consumption. From communication theory (as we will see in section 3.2.2), given a modulation scheme, the higher the modulation parameter, the higher the number of bits per symbol used in data transmission and therefore the lower the transmission time required. Also, if the modulation parameter increases, the energy spent increases as well. Therefore there exists a direct relationship between energy and latency (transmission time). It can be observed in Fig. 2.12.

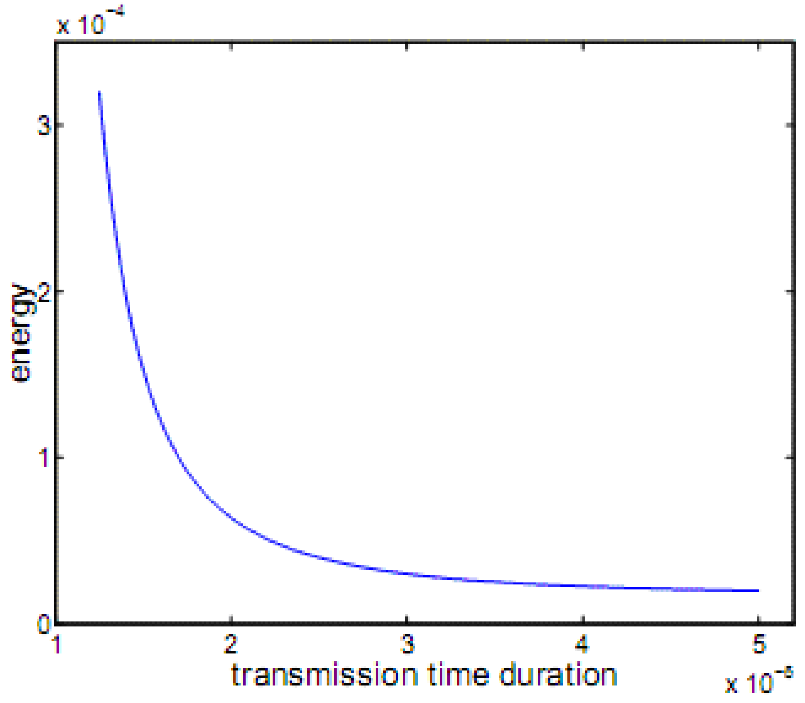

Figure 2.12: Energy-delay trade-off in DMS

Several approaches have used this property of communication's theory in many different ways. In order to give the reader a better comprehension about this mechanism, some relevant works are shown in the following:

Curt Schurgers and Mani B. Srivastava [13, 14]

They presented a novel technique called modulation scaling for wireless sensor networks. From communication's theory they found a dependency between the energy and the modulation parameter b in a QAM scheme

$$E_{bit} = \left[C_s \cdot \left(2^b - 1\right) + C_E + C_R\right] \cdot \frac{1}{b} \tag{2.4}$$

Where C_S is a parameter that characterizes the power transmission of a node, and C_E and C_R are parameters that characterize the electronic power of a node. Considering that the transmission time of one packet is inversely proportional to the modulation parameter (b_i) and the symbol rate (R_i)

$$\tau_i = \frac{s}{b_i \cdot R_i} \tag{2.5}$$

The relationship between energy (to transmit a packet) and latency was

$$E_i = \left[C_s \cdot \left(2^{\frac{s}{\tau_i \cdot R_i}} - 1\right) + C_E + C_R\right] \cdot \tau_i \cdot R_i \tag{2.6}$$

The plot of this function can be observed in Fig. 2.12. Therefore, based on the above relationship, Schurgers and Srivastava proposed to add a power management module to the radio for real-time traffic. The basics of their proposal were the following ones:

- Each real-time stream is modelled as a tuple (L_k, T_k) where L_K is the maximum packet size and T_k the time period.
- The periods and packet size can be different for the different streams.
- Scheduler based on non-preemptive Earliest Deadline First (EDF).

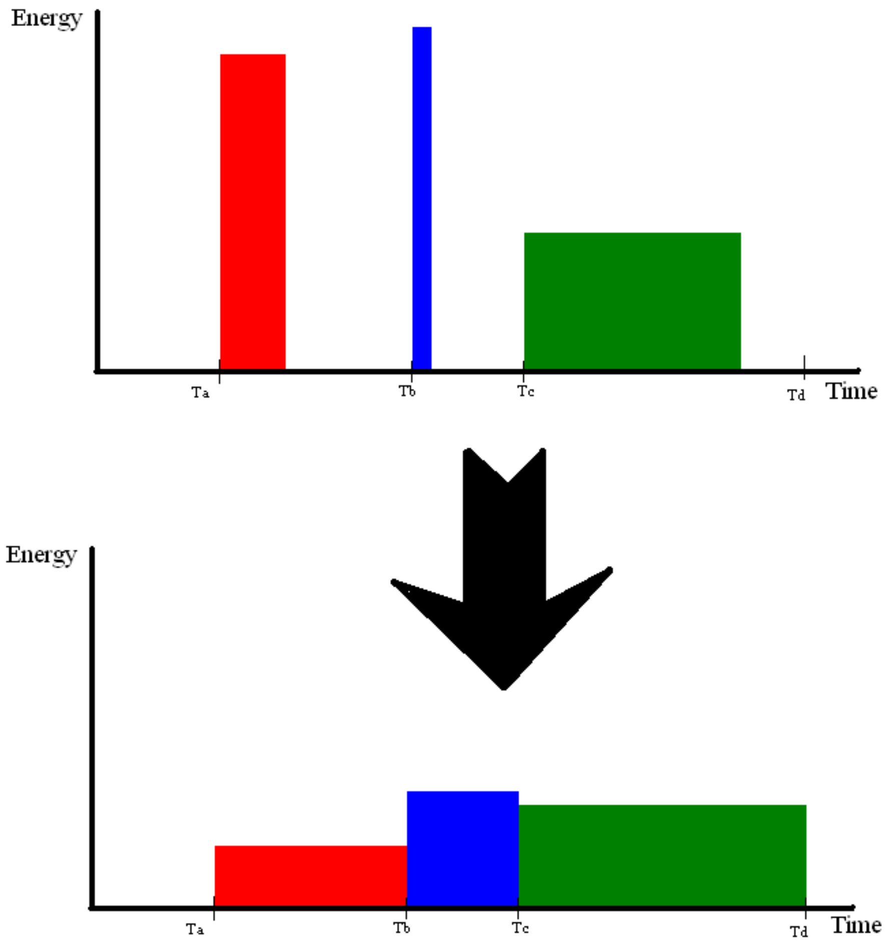

Figure 2.13: Before and after applying scaling factors

The algorithm was composed of two steps:

1. Admission step: Assuming that all packets are maximum size, when a new stream is admitted to the system, a static scaling factor is computed (α_{static}). This factor is the minimum possible such that if the modulation setting for each packet would be scaled by it, the schedulability test is still satisfied. In other words, it computes the slowest transmission speed at which all the packet streams are schedulable.

2. Adjustment step: At run-time, packets are scheduled using EDF. Before starting the transmission, the actual size of each packet is known. Therefore a new scaling in performed such that each packet adapts its transmission time to the maximum transmission time allowed. The factor used in this step is called $\alpha_{dynamic}$. Finally, if the deadline of some packet is earlier than the arrival time of the next one, another adaptation can be performed with $\alpha_{stretch}$.

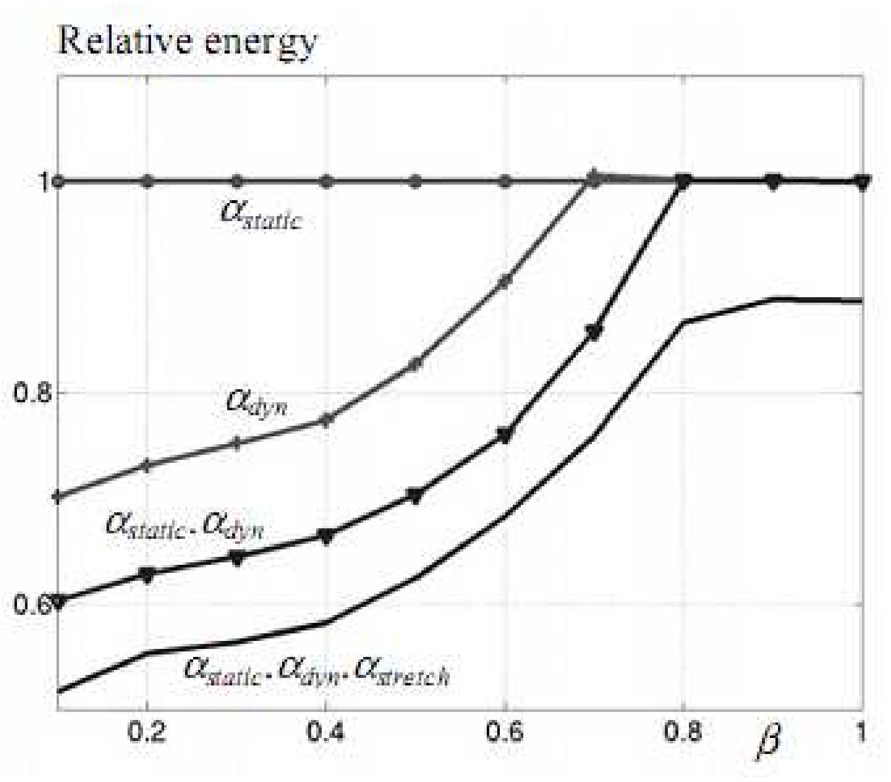

Figure 2.14: Energy performance. Reprinted from [14].

The scheduler combines all three scaling factors and applies to the modulation factors used for transmitting each packet. As it can be seen in Fig. 2.13 after performing the scaling, the link utilization is higher due to the slowing down of transmission in order to achieve less energy consumption. The performance of the algorithm (Fig. 2.14) showed that important energy savings could be reached when using all three scaling parameters and the packet size variation increases (β decreases):

Loading in time [15]

Again, same authors, Schurgers and Srivastava presented another mechanism based in modulation scaling where channel condition variations were taken into account. The channel was modelled as Rayleigh fading channel and the normalized channel power-gain factor α was used in order to compute the modulation parameter at each instant of time. The parameter α was obtained through channel estimation and was updated at a rate f_{update}. Between estimates of α, modulation level was kept constant. In Fig. 2.15 can be seen how depending on de value of α, different modulation parameter is selected. The mapping between each α_i to its modulation parameter b_i is carried out using the following equation

$$b_i = 2\left\lfloor 1 + \frac{1}{2} \cdot \log_2(\alpha_i) - \frac{1}{2} \cdot \log_2(d_1) \right\rfloor \tag{2.7}$$

Simulations showed that a reduction of energy consumed during transmissions was achieved. An example of a file transfer is showed on Table 2.2.

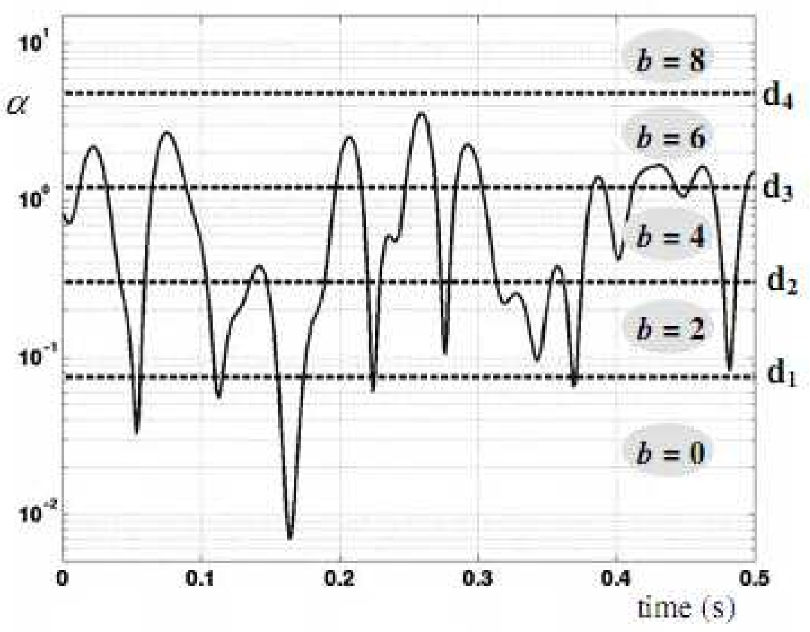

Figure 2.15: Relationship between thresholds and modulation. Reprinted from [15].

Maximum throughput	Fixed throughput	Loading in time
$E_{tot} = 341$J	$E_{tot} = 133$J	$E_{tot} = 71$J

Table 2.2: Energy savings

Zongkai Yang et al. [16]

They developed the concept of modulation scaling applied to WSN in a different way. They considered computing optimal modulation parameters while satisfying end-to-end latency and packet loss constraints. Some assumptions were taken:

- The buffer of each node can hold N packets.
- Packet arrivals follow Poisson process and packet sizes a negative exponential distribution whose mean is s.
- The topology of the network is abstracted as a data aggregation tree.
- Each node is viewed as an M/M/1 queuing system with limited capacity from the viewpoint of queuing theory.

The objective was to find the optimal set of packet service rates $\{u_j\}$ so as to minimize energy consumption. Some constraints had to be satisfied:

- For each source node j, the mean time cost spent on querying and transmitting (W_j) should be less than the latency Γ specified by the application.
- For each source node j, the packet loss rate P_j should be less than the packet loss constraint P_{loss}.

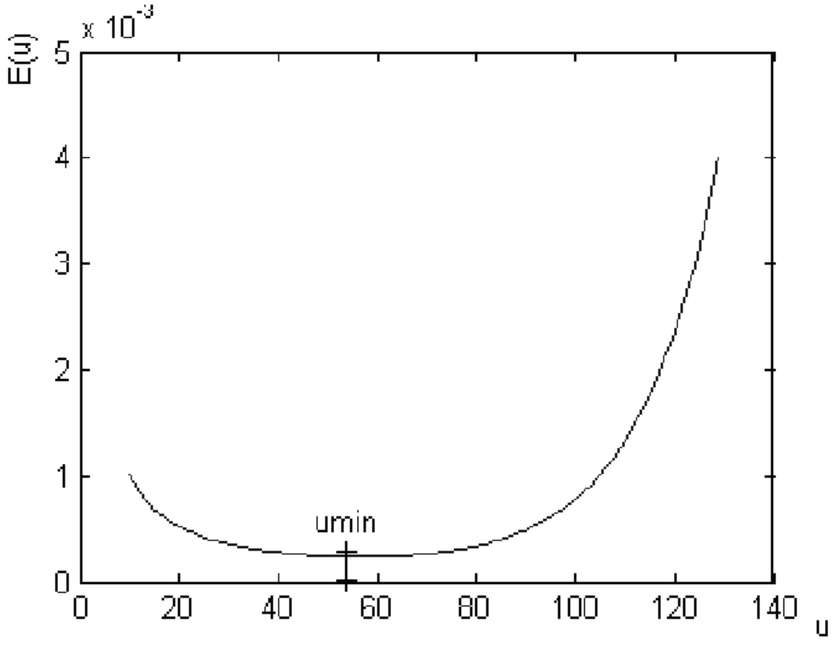

Figure 2.16: Energy function for transmitting one packet. Reprinted from [16]

They found the relationship between the mean energy cost and the packet service rate

$$E(u) = \left[C \cdot \left(2^{\frac{su}{(dk-k+1)R}} - 1\right) + F\right] \cdot \frac{1}{u} \cdot R \tag{2.8}$$

Where k was the aggregation factor and d was the number of children of each node. A plot of the energy consumption of an intermediate node of the aggregation tree is shown in Fig. 2.16.

A centralized management mechanism, based on the constraints mentioned before, was proposed to adaptively adjust the modulation level. They found that important savings could be achieved and also they discovered that the optimal packet service rate of each node was almost proportional to the packet arrival rate.

Yu and Prasanna [17]

They proposed the use of modulation scaling over a multi-hop communication path. Their intuitive objective was based on minimize the maximal energy dissipation over all sensors along the path, with respect to their remaining energy and satisfying a specific end-to-end latency constraint. All their work was focused on QAM scheme. Therefore the energy curves presented by Schurgers and Srivastava were used in their research

$$E_i = \left[C_s \cdot \left(2^{\frac{s}{\tau_i \cdot R_i}} - 1\right) + C_E + C_R\right] \cdot \tau_i \cdot R_i$$

They formulated the problem as follows. Given a series of consecutive single-hop communication links $S_1 \rightarrow S_2 \rightarrow \ldots \rightarrow S_n$, and packet size s; find a feasible schedule of packet transmission $\overrightarrow{\tau}$, so as to minimize the maximal energy dissipation over all nodes

$$OBJ_{\overrightarrow{\tau}} = \max_{i=1}^{n-1} E_i(\tau_i) = \max_{i=1}^{n-1} \left[C_i \cdot \left(2^{\frac{s}{\tau_i \cdot R_i}} - 1\right) + D_i\right] \cdot \tau_i \cdot R_i$$

Their solution was based on the following lemma: A schedule $\overrightarrow{\tau}$ is optimal if and only if satisfies:

1. $\sum \tau_i = T$; and
2. $w_1(\tau_1) = w_2(\tau_2) = \ldots = w_{n-1}(\tau_{n-1})$

where $w\left(\cdot\right)$is the energy function and T the end-to-end latency constraint.

For reaching this solution, they used binary search. Their algorithm was based on computing at the beginning same latency for all nodes and then, performing iterations using binary search between two energy limits until the solution was found. The pseudo code of their algorithm is showed on Fig. 2.17.

A graphical plot is represented (Fig. 2.18) for four nodes. It can be observed that the energy limits correspond to the latency constraint computed at the first step of the algorithm. Also, the optimal solution is plotted, which is between the energy limits and satisfies the lemma mentioned above.

We have to mention that our power-aware algorithm arose from this idea. In fact, one of the steps of our algorithm is computing first the energy-optimality in the same way as has been shown here. In later sections a wide description will be carried out.

```
Begin
1. Set $j = 0$; $\tau_i^j = \frac{T}{n-1}$
2. Calculate $E_{max}^j$ and $E_{min}^j$
3. While $\vec{\tau}^j$ is not optimal, Do
4.     $j = j + 1$; $E_{mid} = \frac{E_{max}^j + E_{min}^j}{2}$
5.     Calculate $\vec{\tau}^j$ by solving equations $w_i(\tau_i^j) = E_{mid}$
6.     If $\sum_i \tau_i^j \in [T(1-\varepsilon), T]$, $\vec{\tau}^j$ is optimal
7.     Else If $\sum_i \tau_i^j < T$
8.         $E_{max}^j = E_{max}^{j-1}$; $E_{min}^j = E_{mid}$
9.     Else
10.        $E_{max}^j = E_{mid}$; $E_{min}^j = E_{min}^{j-1}$
End
```

Figure 2.17: Pseudo code. Reprinted from [17]

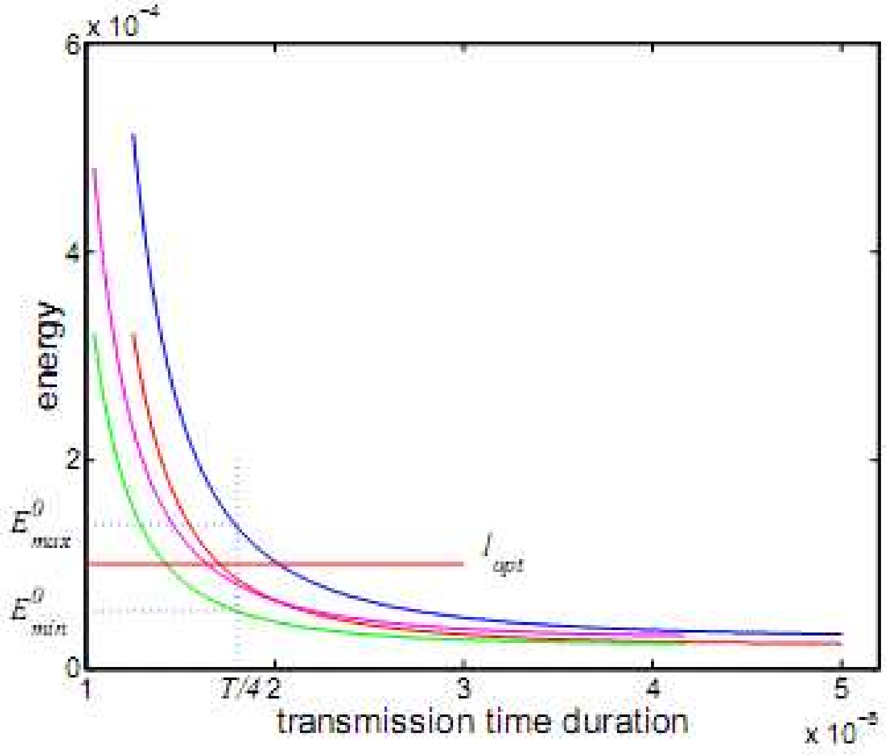

Figure 2.18: Energy functions of the nodes of a path and the optimal solution. Reprinted from [17]

Chapter 3

Problem Definition

Once introduced the reader into the WSN community, we suppose that he is aware that there are many layers in which improvements can be still made. Our aim in this section is to describe the scenario that we adopted for de development of our modulation scaling mechanism. Also, a study of different modulation schemes is performed in order to know which one fits better for our WSN. Furthermore, we describe the mathematical model for the energy used during transmissions between nodes. We perform the demonstration as many authors have carried out [13, 14, 17] and we add our own analysis for special cases. Finally the basis of our approach is revealed to the reader and a justification about the adopted solution is given.

3.1 Scenario

As we saw in previous chapters, one basic feature of WSN is to send information in short distances and forward them many times in order to achieve power efficiency. This geographical model was called multi-hop wireless network (see Fig. 3.1). Due to the different characteristics of each node, as for instance transmission power parameters, electronic parameters, distance to the next node, ...; the energy consumption of each node is going to be different. This means that in a path of nodes, the node with higher consumption of energy will determine the path's lifetime.

Our aim is to develop an algorithm for multi-hop packet transmission that achieves the longest life-time over the all network taking into account the battery energy levels of each node. One way to solve it is by the use of dynamic modulation scaling, where given a modulation scheme, its modulation parameter can vary dynamically depending on the requirements of the system at a given time. In order to solve it, we proposed a scenario with the following characteristics:

- the underlying wireless sensor network is going to be modelled as multi-hop communication paths. In each path, n sensor nodes will be involved (see Fig. 3.1):

  $$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow \cdots \rightarrow S_n$$

  Where $\rightarrow$ represents a simple hop in the path.

- Each node has the possibility of being part of several paths at the same time. Therefore different nodes will have more energy consumption per unit of time than others.
- Each node has several parameters. Most important for our analysis are:
  - Remaining battery level ζ
  - Electronic and transmission power parameters, C_S and C_E
  - Modulation parameter b.
  - Individual transmission time or latency τ_i.
- A packet of size s is going to be transmitted from S_1 to S_n in a given end-to-end latency constraint, denoted by T.
- Each node is assumed to be reliable, and as latency constraints are supposed to be satisfied, we can consider that the buffers are going to be infinite. Thus, packets will not be discarded in our system. It's not the aim of this research to study QoS parameters as for intance packet drops.
- Both transmitting and receiving will be considered to spend same amount of energy when performing the simulations.
- Long-range communications (10 – 100 m.) will be selected for each hop transmission. The reasons are explained in section 3.2.
- For easy analysis, we focus on QAM scheme. However the research made can be easily extended to other modulations schemes as well. In the following section several modulation schemes are commented.

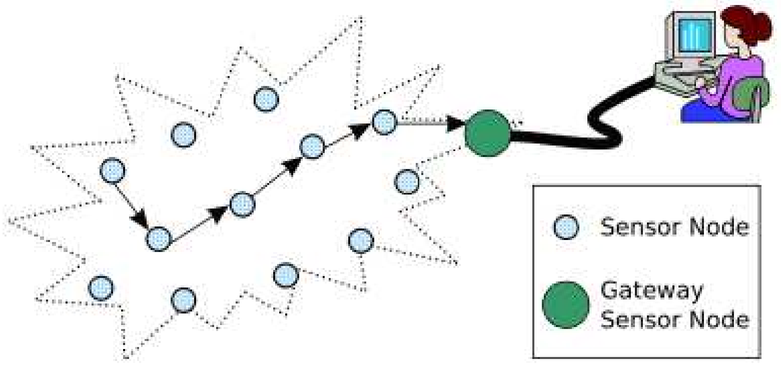

Figure 3.1: Multi-hop wireless sensor network

3.2 Energy Model

3.2.1 Modulation in Wireless Sensor Networks

Several digital modulations can be used in different protocols for wireless networks. Most used are PSK, QAM, FSK, OFDM, etc... In the following, we are going to make a short comparison between some important modulations and we are going to choose one of them to develop an energy model for our research.

First of all, OFDM modulation can be discarded because it requires expensive transmitter circuitry giving poor power efficiency. It is more appropriate for protocols that are awareness of power consumption. The digital modulation methods named before can be compared in a number of ways. For example, one can compare them on the basis of the SNR required to achieve a specified probability error as long as we have defined a data rate of transmission or, equivalently, a fixed bandwidth. Also, it can be represented the normalized data rate (R/W) versus the SNR required to achieve a fixed error probability. In the following several plots are presented in order to determine which modulation fits better into our objectives.

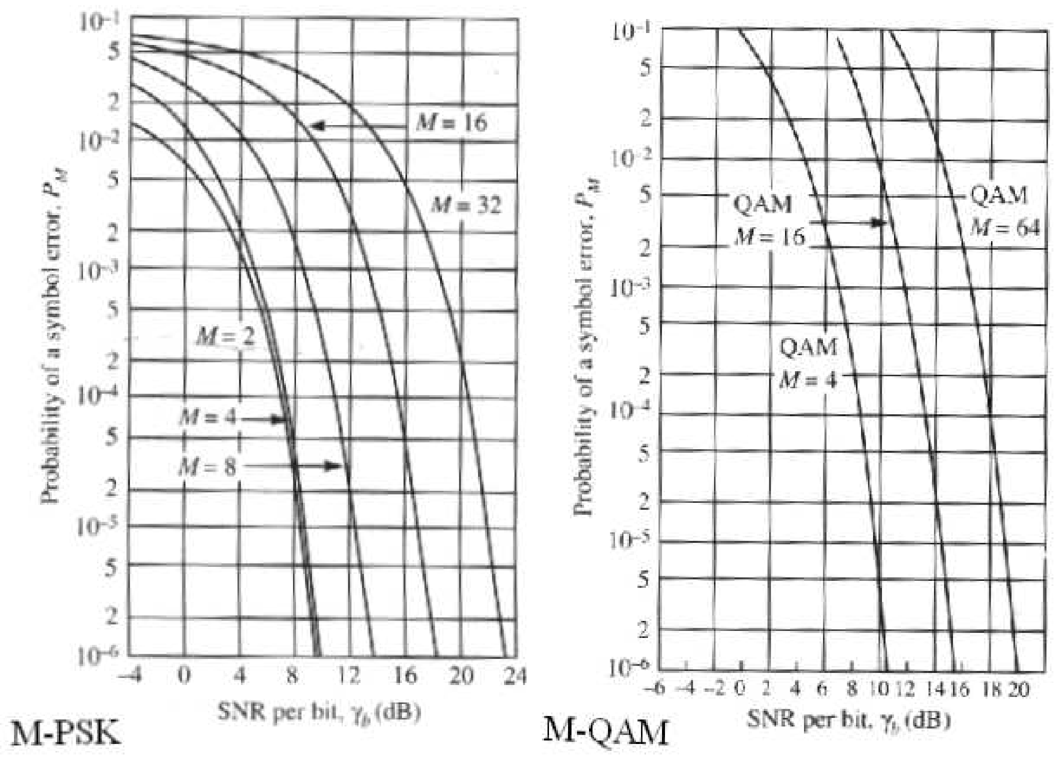

Figure 3.2: Prob. of symbol error vs. SNR for M-PSK and M-QAM. Reprinted from [18].

For M-PSK and M-QAM we can observe the relation between the probability of a symbol error and the SNR per bit in Fig. 3.2. From [18] we know that for M-ary PSK the probability of a symbol error is approximated as

$$P_m \simeq 2 \cdot Q\left(\sqrt{2 \cdot \text{SNR}} \cdot \sin \frac{\pi}{M}\right) \tag{3.1}$$

and for M-ary QAM is approximated as

$$P_m \simeq 1 - \left(1 - 2 \cdot \left(1 - \frac{1}{\sqrt{M}}\right) \cdot Q\left(\sqrt{\frac{3}{M-1} \cdot \text{SNR}}\right)\right)^2 \tag{3.2}$$

Since the error probability is dominated by the argument of Q function, we simply compare the arguments of Q of both modulations

$$R_m = \frac{{}^{3}/_{M-1}}{2 \cdot \sin\left({}^{\pi}/_{M}\right)^2} \tag{3.3}$$

For $M = 4$ we have that $R_m = 1$ and therefore 4-PSK and 4-QAM yield comparable performance for the same SNR per symbol. When $M > 4$ we find that $R_m > 1$, so that M-ary QAM yields better performance than M-ary PSK.

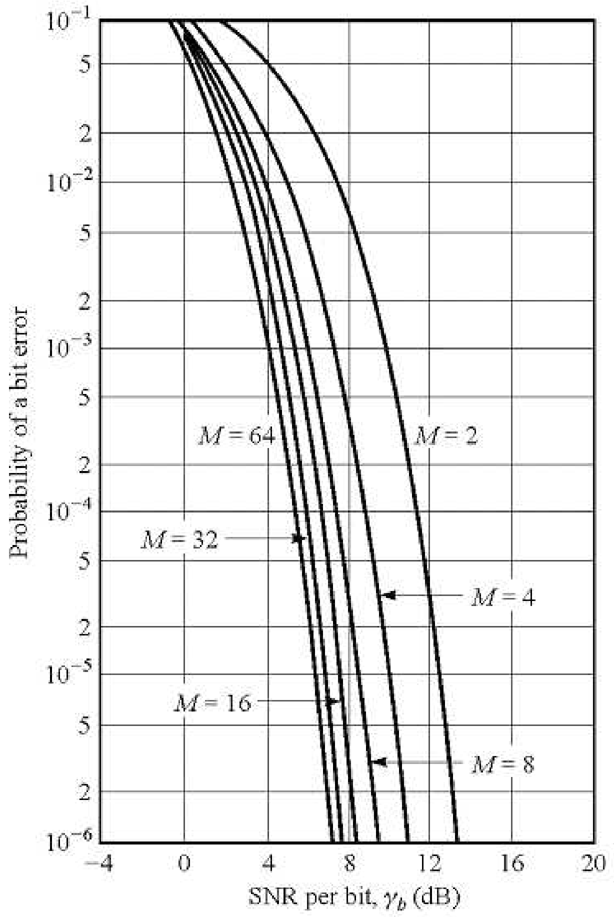

Figure 3.3: Prob. of bit error vs. SNR for M-FSK. Reprinted from [18].

On the other hand for M-FSK the relation between the probability of a bit error and the SNR can be observed in Fig. 3.3. It is known [18] that bit error probability (P_b) in M-FSK is related to symbol error probability (P_m) as

$$P_b \simeq \frac{P_m}{2} \tag{3.4}$$

Therefore, it can be observed that M-ary modulation is more energy-efficient if we use M-FSK. For example for $M = 64$, $P_m = 10^{-6}$; $\mathrm{SNR}_{\mathrm{MQAM}} \simeq 20\,\mathrm{dB}$ while $\mathrm{SNR}_{\mathrm{MFSK}} \simeq 7\,\mathrm{dB}$. As we will describe later, the transmitted power is going to be dependent of the SNR. The more SNR required for a specified P_b, the more power consumption. An example [19] of transmit power versus bandwidth efficiency ($^R/_W$) is plotted for a wireless microsensor system in Fig. 3.4.

The power required for FSK increases slowly compared to PSK or QAM. The sacrifice, however, is bandwidth, which uses to be limited in wireless sensor networks. For instance, 8-FSK uses four times as much bandwidth as M-PSK. The problem can be solved by careful planning of the spectrum. In the unlicensed band in the GHz regime, large bandwidth is available to make M-FSK a realistic option. However, it is not the aim of our thesis to explore this issue.

Therefore, QAM modulation is going to be used due to its easy of implementation and analysis [18]; it is more efficient than PSK and not requires taking care about the bandwidth as in FSK. However, the future techniques developed can be perfectly extended to other modulations just making a similar analysis like the one that comes in the following.

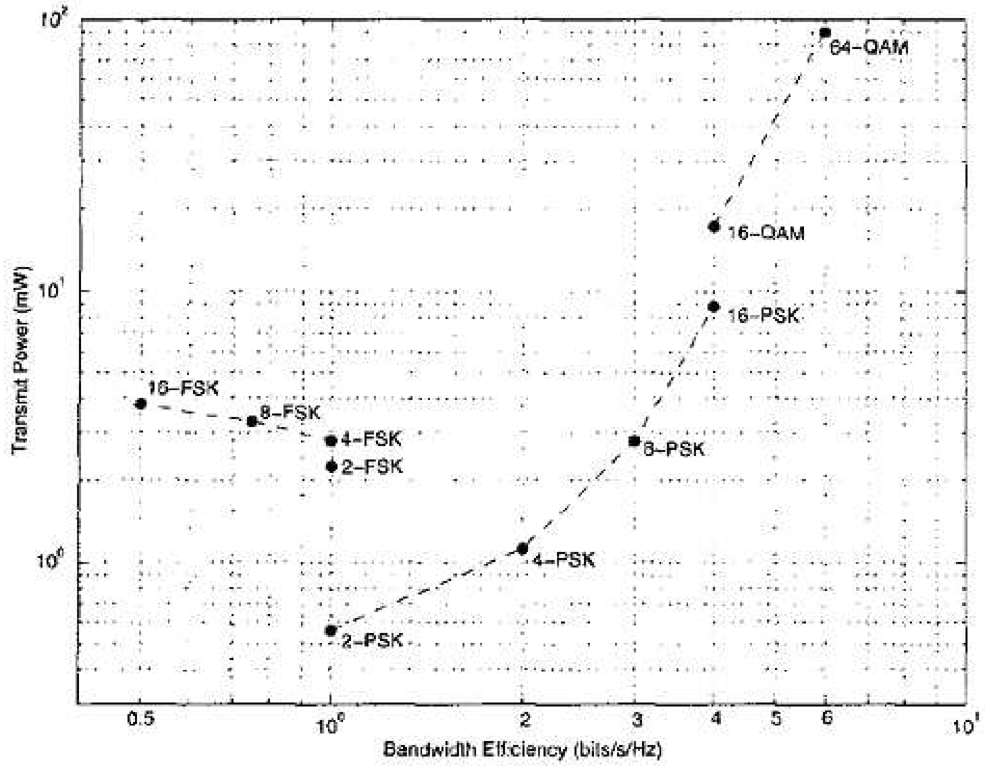

Figure 3.4: Transmit power vs. bandwidth efficiency in fading channel. Reprinted from [19].

3.2.2 The Energy Model

First, we need to derive the relevant expressions. For QAM modulation, the performance in term of Bit Error Rate is given by [18]

$$\text{BER} = \frac{4}{b} \cdot \left(1 - \frac{1}{2^{b/2}}\right) \cdot Q\left(\sqrt{\frac{3}{2^b - 1} \cdot \frac{\varepsilon_{av}}{N_0}}\right) \tag{3.5}$$

where b is the number of bits per symbol and $\frac{\varepsilon_{av}}{N_o}$ is the average signal to noise ratio (SNR) per symbol.

On the other hand, we know that

$$\text{SNR} = \frac{P_s}{P_n} \cdot A \tag{3.6}$$

where P_s is the transmit power delivered mainly in the power amplifier, A is a factor that contains all transmission loss components and P_n is the noise power whose function comes given by

$$P_n = N_0 \cdot \beta \cdot R_s \tag{3.7}$$

where N_0 is the noise power spectral density, β takes into account other elements such as filter non-idealities, and R_s is the symbol rate.

If we manipulate the equations

$$P_S = \frac{\text{SNR}}{A} \cdot P_n = \frac{\varepsilon_{av}/N_0}{A} \cdot N_0 \cdot \beta \cdot R_s \tag{3.8}$$

From 3.5 and 3.8 we obtain that

$$\text{SNR} = \frac{\varepsilon_{av}}{N_0} = \frac{1}{3} \cdot (2^b - 1) \cdot \left[Q^{-1}\left(\frac{b}{4} \cdot \text{BER} \cdot \left(1 - \frac{1}{2^{b/2}}\right)^{-1}\right)\right]^2 \tag{3.9}$$

and therefore from 3.8 and 3.9

$$P_S = \frac{N_0 \cdot \beta}{A} \cdot \frac{1}{3} \cdot \left[Q^{-1}\left(\frac{b}{4} \cdot \text{BER} \cdot \left(1 - \frac{1}{2^{b/2}}\right)^{-1}\right)\right]^2 \cdot \left(2^b - 1\right) \cdot R_s \tag{3.10}$$

Our aim is to study the power consumption against the modulation factor b. Thus, we can define a new constant C_S. It will be a parameter to characterize the power transmission of a sensor node. So, we can express C_S as

$$C_S = \frac{N_0 \cdot \beta}{A} \cdot \frac{1}{3} \cdot \left[Q^{-1}\left(\frac{b}{4} \cdot \text{BER} \cdot \left(1 - \frac{1}{2^{b/2}}\right)^{-1}\right)\right]^2 \tag{3.11}$$

Note that C_S has a weak dependence of b which is inside the inverse function of $Q(\cdot)$. Also the BER parameter is going to be fixed to a target value as in many practical scenarios. Therefore, C_S can be considered a constant.

Thus far, we have that transmit power can be written as

$$P_S = C_S \cdot \left(2^b - 1\right) \cdot R_s \tag{3.12}$$

On the other hand, transmit power is not the only source of power spending. Electronic circuitry for filtering, modulating, up-converting, etc... contributes as well. Some areas of the circuitry are going to operate at frequencies that are dependent of symbol rate and follow the variations of it. Other ones are going to be dependent of a fixed rate. Thus we have two parameters that incorporate the proportionality factors and switching activity, they are C_A and C_B. Therefore, from [13] we can state that the electronic power comes determined by

$$P_E = \left[C_E + C_R \cdot \frac{R_{s_{max}}}{R_s} \right] \cdot R_s \tag{3.13}$$

$$C_E = C_A \cdot V^2 \; C_R = C_B \cdot V^2 \tag{3.14}$$

Where V is the voltage at which the system works.

We have defined the transmission power and the electronic one. Now, summing both contributions is enough in order to compute the total power consumption. However it makes more sense to look at the energy consumption rather than the total power. Therefore we can describe the energy to transmit one bit as

$$E_{bit} = (P_S + P_E) \cdot T_{bit} \tag{3.15}$$

$$T_{bit} = \frac{1}{b \cdot R_s} \tag{3.16}$$

And if we put equations 3.12 and 3.13 inside 3.15 we obtain that

$$E_{bit} = \left[C_S \cdot \left(2^b - 1 \right) + C_E + C_R \cdot \frac{R_{s_{max}}}{R_s} \right] \cdot \frac{1}{b} \tag{3.17}$$

It is clear that operating at the maximum symbol rate is preferable for any b due to the fact that higher symbol rates result in both a lower T_{bit} and a lower E_{bit} [13]. Hence, 3.17 can be expressed as

$$E_{bit} = \left[C_S \cdot \left(2^b - 1 \right) + C_E + C_R \right] \cdot \frac{1}{b} \tag{3.18}$$

As we described in the scenario, a packet of size s is to be transmitted along the path. Therefore the time duration of the transmission of one packet can be modelled as

$$\tau_i = \frac{s}{b \cdot R_s} \tag{3.19}$$

And therefore the energy to transmit one packet of size s, E_i, is defined as

$$E_i = \left[C \cdot \left(2^b - 1 \right) + D \right] \cdot \frac{s}{b} \tag{3.20}$$

which also can be described as

$$E_i = \left[C \cdot \left(2^{\frac{s}{\tau_i \cdot R_s}} - 1 \right) + D \right] \cdot \tau_i \cdot R_s \tag{3.21}$$

3.2.3 Energy model for odd values of k

The energy model for an M-QAM modulation calculated before is exact when even values of b ($M = 2^b$) are stated. We followed the same way as Shurgers and Srivastava [13, 14] to describe the last model. However, nobody explains what happens if the modulation parameter is odd.

When b is even, rectangular signal constellations are used due to the easy of implementation of modulators and demodulators. Although they are not the best solution when $M \geq 16$, the average transmitted power required to achieve a given minimum distance is only slightly greater than the average power required for the best M-ary QAM signal constellation.

Furthermore, when b is odd best solutions use to be circular multi-amplitude constellations. However, as [18] mentions, it is rather easy to determine the error rate for rectangular signal set despite circular constellations are more efficient. It is relatively straightforward to show that the symbol error probability is tightly upper-bounded as

$$P_m \leq 1 - \left(1 - 2 \cdot Q\left(\sqrt{\frac{3}{M-1} \cdot \mathrm{SNR}}\right)\right)^2 \leq 4 \cdot Q\left(\sqrt{\frac{3}{M-1} \cdot \mathrm{SNR}}\right) \tag{3.22}$$

Making an analogue study as for equation 3.11, we can achieve that

$$C_s = \frac{N_0 \cdot \beta}{A} \cdot \frac{1}{3} \cdot \left[Q^{-1}\left(\frac{b}{4} \cdot \mathrm{BER}\right)\right]^2 \tag{3.23}$$

and therefore we can state for Cs that

$$C_{even\,b} = \frac{N_0 \cdot \beta}{A} \cdot \frac{1}{3} \cdot \left[Q^{-1}\left(\frac{b}{4} \cdot \mathrm{BER} \cdot \left(1 - \frac{1}{2^{b/2}}\right)^{-1}\right)\right]^2 \simeq C_{odd\,b} = \frac{N_0 \cdot \beta}{A} \cdot \frac{1}{3} \cdot \left[Q^{-1}\left(\frac{b}{4} \cdot \mathrm{BER}\right)\right]^2 \tag{3.24}$$

due to the weak dependence of Bit Error Rate and modulation parameter b that are inside the function $Q^{-1}(\cdot)$.

Hence, this analysis shows that equation 3.21 is also a good approximation for odd values of the modulation parameter. With this last remark, we have finished the energy model for our modulation mechanism.

3.3 Our Approach

The solution adopted by Yu and Prasanna [17], which relied on minimizing the energy dissipation over all sensors along a path spending same energy in all the nodes, is not optimal due to, as we described in our scenario, different nodes can have more traffic load than others. Therefore the node's consumption of energy inside a path can be different to the consumption of the other nodes.

Our approach tries to balance the difference in power consumption of the nodes. In a determined time, we are going to have a specified latency, modulation parameters for each node and also different remaining battery levels. Therefore, our mechanism is based on transmitting faster in those nodes with more remaining energy, and transmitting slowly in those nodes with lower battery levels. Transmit faster means to have a lower delay and therefore consume more energy. The idea is to steal to the "rich" node transmission time in order to give it to the "poor" node. Thus, nodes with high energy will spend a little bit more energy and low energy nodes will save energy. This concept let us to increase the lifetime in a path of nodes.

Therefore, given a path of n nodes (and therefore $n-1$ hops), two main constraints are suggested

$$\sum_{i=1}^{n-1} \tau_i = T \tag{3.25}$$

$$w(\tau_1, \zeta_1) \geq w(\tau_{n-1}, \zeta_{n-1}),\ w(\tau_2, \zeta_2) \geq w(\tau_{n-2}, \zeta_{n-2}), \ldots \\ \text{with}\, \zeta_1 \geq \zeta_2 \geq \zeta_3 \geq \zeta_4 \geq \ldots \geq \zeta_{\mathrm{n-1}} \tag{3.26}$$

As we mentioned in the scenario definition, τ_i is the individual transmission time of a node. ζ_i is the remaining battery level and T denotes the end-to-end latency. The function $w(\tau_i, \zeta_i)$ denotes the energy consumption of a node as a function of latency and remaining energy. Hence, first equation is related to satisfying the path latency constraint. The second equation tries to balance the energy consumption between nodes. As it can be seen, the balance is performed with pairs of nodes where the selection of the pairs is not random. Assuming that nodes are put in order form higher energy to lower one, the methodology to generate pairs follows the next formula

$$\mathrm{node}(x) \Longleftrightarrow \mathrm{node}(n-x)$$

Our methodology does not take care about spending the minimum energy along all nodes. It centres all its efforts in maximizing the lifetime despite the fact that it consumes more energy than required. This over spending in energy is justified with the following fact. The lifetime of a path is computed as the time from the beginning of the path until some node dies. Therefore, at this time, several nodes can still have energy in their batteries, energy that is wasted. Our protocol tries to use that wasted energy by making over-spending of energy in high battery nodes in order to achieve savings in low battery ones.

Thus far, we only have been thinking in the physical layer. From this point of view, our algorithm is optimal. However, from the network or application layer view we can realise that not always our mechanism is efficient and useful. For instance, our methodology is not designed to be working with dynamic multi-hop routing. An example of this kind of routing was described in section 2.2.1 in page 18 . In this kind of routing, the lifetime cannot be described as we did before. When a node dies, the remaining battery of the rest of the nodes of a path can be still used, because the routing protocol selects a neighbour node to replace the death one. However, this management of nodes imposes having higher radio coverage for each node in order to select newer ones in case of death. Also this techniques use to be applied when a dense-deployment of the nodes is carried out. Therefore, our protocol is designed for networks with manually-deployment where the radio coverage can be static and relatively shorter, and where defined structures can be generated, such as tree topologies, clustered topologies with static routing, mesh based topologies, and so on.

In addition, our approach is for long-range communications (hops from 10 to 50 metres). The reason comes from a study of the energy curves performed, where as it can be observed in Fig. 3.5 and

Fig. 3.6, it is difficult to achieve energy savings in short range communications.

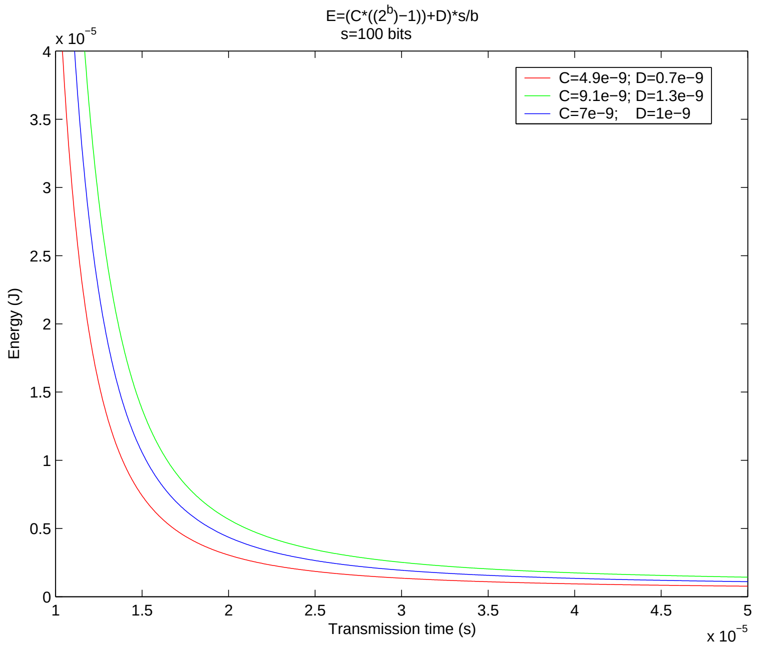

Figure 3.5: Energy curves for long-range communications

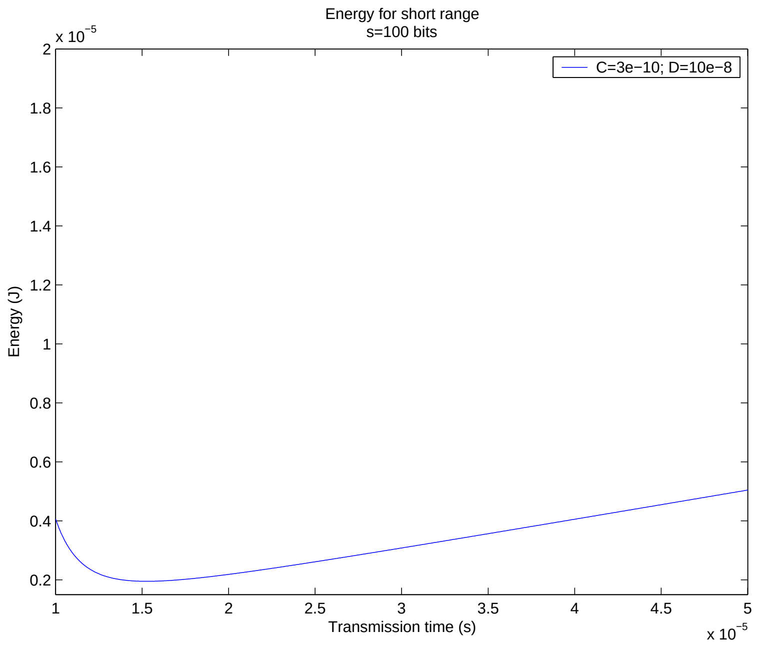

Figure 3.6: Energy curves for short-range communications

The energy savings of a transmission in long range communications can be seen in fig. 3.7 for different latency constraints. If we call the node with more remaining energy node A, and the node with less energy node B, we can observe that if we spend 150% energy in node A, it can be saved from 30% to 15% of energy in node B. Although they are not good savings, in combination with several nodes allow to achieve great savings. In practice, the implementation of our mechanism is based on selecting dynamically the number of nodes required to achieve a specified percentage of savings in node B. It will be explained in the following chapter.

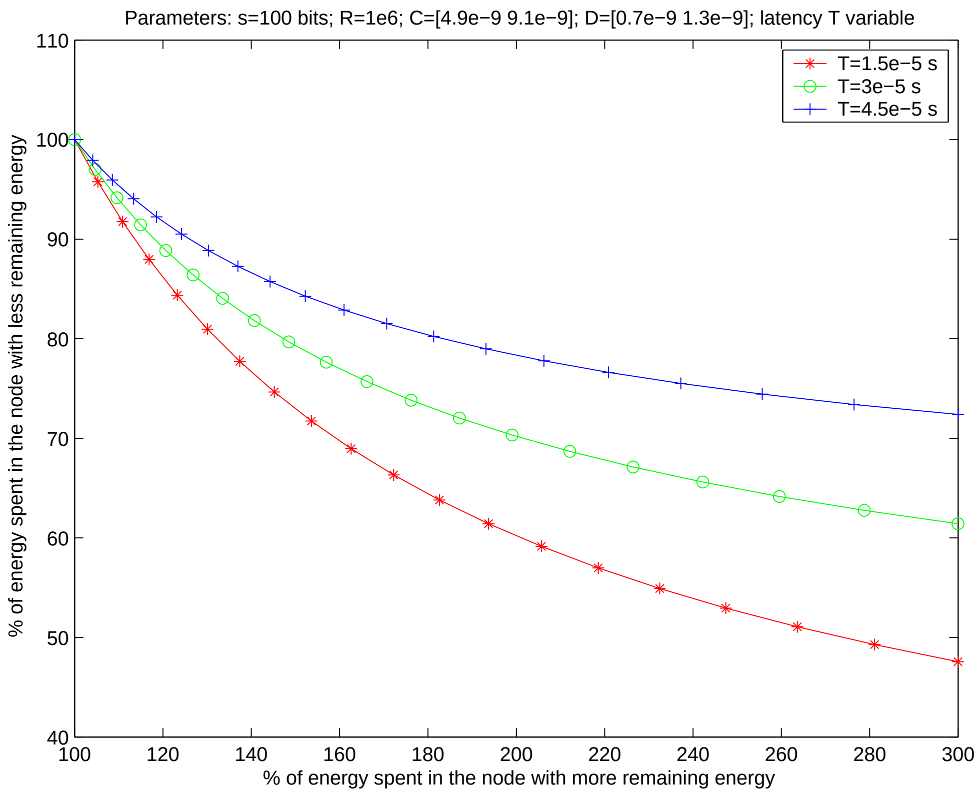

Figure 3.7: Relationship between the overspending in the node with more remaining energy and the saving in node with less energy

Note that the savings showed before have been considering that the transmission time of a node can be changed in a continuous range of time. However, due to the fact that transmission time is directly bounded to the modulation parameter, and modulation parameter only can adopt an integer value, the range of time has discrete values. This means that we are limited to take use of that transmission times that are computed from different values of the modulation parameter as showed in Fig. 3.8, where a plot with dots for the possible discrete values of the modulation parameter is shown.

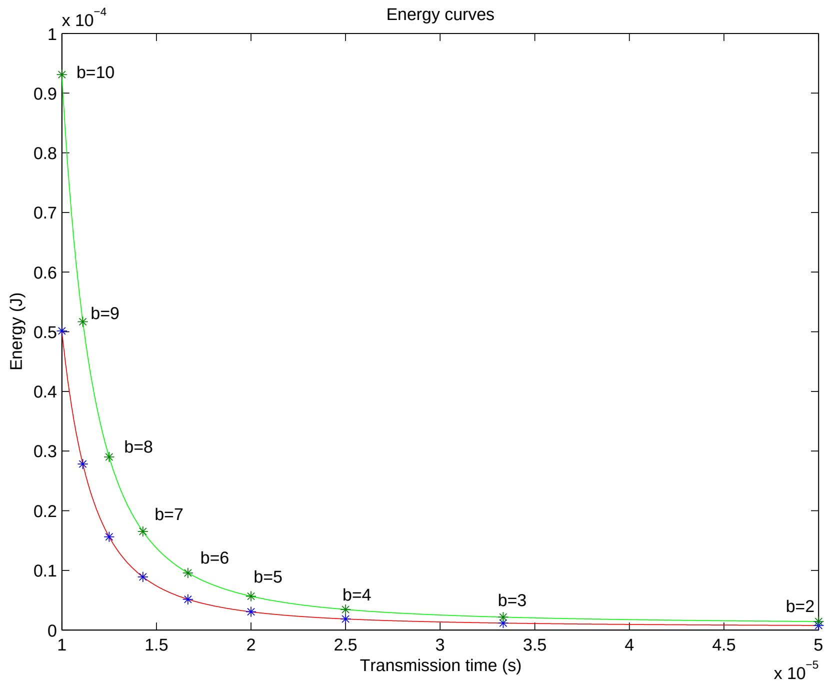

Figure 3.8: Discrete points (E_i, τ_i)in an energy curve

Finally, mention that the simplifications and assumptions taken for the energy model were based on the same ones taken in the papers that work with modulation scaling [13, 14, 15, 16, 17]. Therefore, our results can be perfectly compared with the results of these different techniques due to we are not performing our own simplifications and therefore we are not giving some kind of advantage to our algorithm. Furthermore we have to comment that the parameters used for the simulations are the same parameters that [17] takes into account. Therefore, the comparisons that we made in regard to this paper are completely valid.

Chapter 4

Proposed Solution

By now the reader should know the main issue that tries to cover our approach. The higher traffic loads in some nodes of a path shortens the lifetime of the network. This is because of the multiple paths that a node can be involved in. Therefore our solution tries to manage the modulation parameter of nodes in order to reach savings in those ones that consume more energy. In order to perform it, a centralized algorithm was developed. The idea is to perform all the computations in the sink, and then forward the results to the other nodes. It is feasible only if in upper layers, the current lifetime and modulation parameters of each node are added to the packets being sent, in order to provide to the sink the required information. This centralized solution may seem that it overloads the network with extra packets for the update of the nodes, but it is not in that way due to it is not our aim to execute our algorithm each transmission, but it is to run it each specified number of transmissions or time. For instance, in our simulations, we run the algorithm each 100 hundred transmissions.

In the following, the structure of our solution will be described as well as each of the algorithms developed.

4.1 Structure

Our solution is composed of a main file, called *main.m*, which performs the management of other files in order to reach the solution of all modulation parameters for each node of a path. *Main.m* manages sub-algorithms such as *balance_b.m* and *optimal_b.m*, who are the responsible of the correct assignment of the modulation parameters. The dependence between all files involved in our protocol is represented here:

- main.m:
    - energy_model2.m
    - energyconstraints.m
    - balance_b.m
    - optimal_b.m:
        - optimal_energy.m:

· energy_model.m

· fixedpoint.m

· energy_fixedpoint.m

4.2 Description of Sub-files

We need to describe first all the files that *main.m* uses in order to be able to explain the main algorithm. The code of each m-file can be read in the appendix A.

4.2.1 Fixedpoint.m

[x,iter,z,tcl,tcc]=fixedpoint(g,x0,tol,maxiter,C,D,Eopt,s)

This function allows us to solve the equation of the energy spent by a node when we know the energy but not the latency

$$E_i = \left[C \cdot \left(2^{\frac{s}{\tau_i \cdot R_s}} - 1\right) + D\right] \cdot \tau_i \cdot R_s \Longrightarrow \tau_i??$$

The method used is the fixed point algorithm in order to reach the solution. *Fixedpoint.m* uses the function *energy_fixedpoint.m*, where the energy model is stored in the way $x = g(x)$. In addition, *fixedpoint.m* allows us to determine the precision of the solution through tol parameter.

4.2.2 Energy_model.m

E=energy_model(C,D,R,s,t)

It contains the mathematical expression of the energy spent by a node

$$E_i = \left[C \cdot \left(2^{\frac{s}{\tau_i \cdot R_s}} - 1\right) + D\right] \cdot \tau_i \cdot R_s$$

Its inputs are the transmission power parameter of a node, C; the electronic parameter of a node, D; the symbol rate R, the data packet size s and the transmission time t. From all of them, it computes the energy used for one transmission of a packet of size s. Note that *energy_model2.m* computes the same but instead of using transmission time, it uses modulation parameter.

4.2.3 Energyconstraints.m

[ome,gam]=energyconstraints(energy,max_energy)

This file was designed for computing the proper energy constraints depending on the energy battery level of each node. We have two constraints:

- Omega parameter, *ome*, describes us the amount of energy that we want to save in nodes with less battery.
- Gamma parameter, *gam*, describes the amount of energy that we want to spend in nodes with high battery level.

This file is really important because it determines the behavior of *main.m*. It can manage static constraints, this means, same parameters during all the life of a node, or it can compute different constraints (dynamic) taking into account the battery level of each node.

4.2.4 Optimal_energy.m

[Eopt,Tn]=optimal_energy(T,n,C,D,R,s,PR)

This function is based on the work done by Yu and Prasanna [17]. Given a group of nodes the aim is to find a group of points $(E(i), \tau(i))$ where the following constraints must be satisfied:

- $E(i) = E(i+1),\ i = 0, 1, \ldots, n-1;$
- $\sum(\tau_i) = T;$

This means that we have to find a solution where all nodes consume the same amount of energy and where the latency constraint must be satisfied. From the point of view of the total energy spent, this solution is the optimal one. As Yu and Prasanna proposed, it is carried out with the use of binary search based approximation algorithm (see section 2.2.3 on page 29).

4.2.5 Optimal_b.m

[b,Eopt,b0,E0,E_S,Tn]=optimal_b(T,C,D,R,s)

The value of b must be rounded, because as we know, the modulation parameter must be integer. Therefore the aim of *optimal_b.m* is to round the optimal modulation parameters in order to achieve lower increments of b in nodes with less remaining energy. The rounded modulation parameters are given in the output parameter b.

The most important input parameters are C and D. They are, respectively, vectors containing the transmission power and electronic power parameters of a path of nodes. The lower index in these vectors, the higher the remaining battery level (ζ_i). Thus, $\zeta_1 > \zeta_2 > \zeta_3 > \ldots > \zeta_i$.

The diagram flow of the algorithm can be observed in Fig. 4.1. It starts computing the exact optimal modulation parameters for each node of the path. Then a rounding is performed over each node. If latency constraint is satisfied the optimal values of modulation parameter are ready for their use in upper level algorithms. If not, we go into a loop. It is based in the following considerations:

- As the lower index of the node the higher is the remaining battery, we should increase in one unit the modulation parameter of the lower index nodes first.
- When we have performed the first rounding, some nodes have rounded towards infinite and other ones towards zero. So our consideration is to increase first the nodes that rounded towards zero ($b_{i_t} < b_{i_{(t-1)}}$) than that ones that rounded towards infinite. Thus, we can achieve the given latency constraint using less energy. Note that $j(i) = 0$ at the beginning of the loop.
- The priority for the first consideration is higher to the priority of the second one. Therefore it has to be evaluated the first consideration before the second one.

For more details see implementation in appendix A.

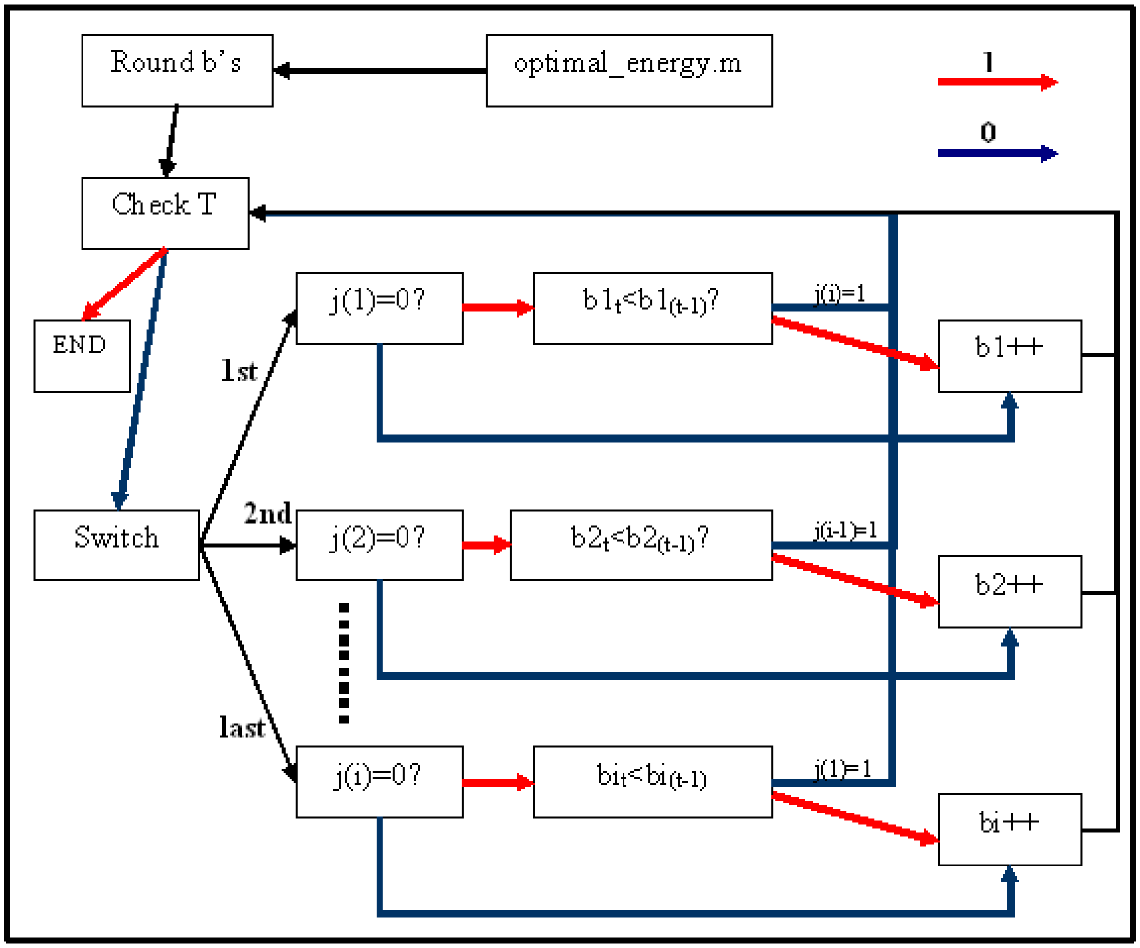

Figure 4.1: Flow diagram of optimal_b.m

4.2.6 Balance_b.m

[bb,energy_spent,Ebalanced,Tn]=balance_b(b,E0,E_S,omega,C,D,T,R,s)

Given a group of nodes (here after *balancing group*), the aim of *balance_b.m* is to find the proper modulation parameters for balancing group nodes in order to achieve an energy saving in the node selected as low battery level node (here after *target node*) while satisfying a latency constraint.

The most important inputs are the optimal modulation parameters (calculated by *optimal_b.m*) and Ω (given by *energyconstraints.m*). As we said before, Ω is the amount of energy that we want to spend in the target node. It is comprised between 0 and 1. For instance, $\Omega = 0.63$ would be a 63% of energy saving.

The diagram flow of the algorithm can be seen in Fig. 4.2.

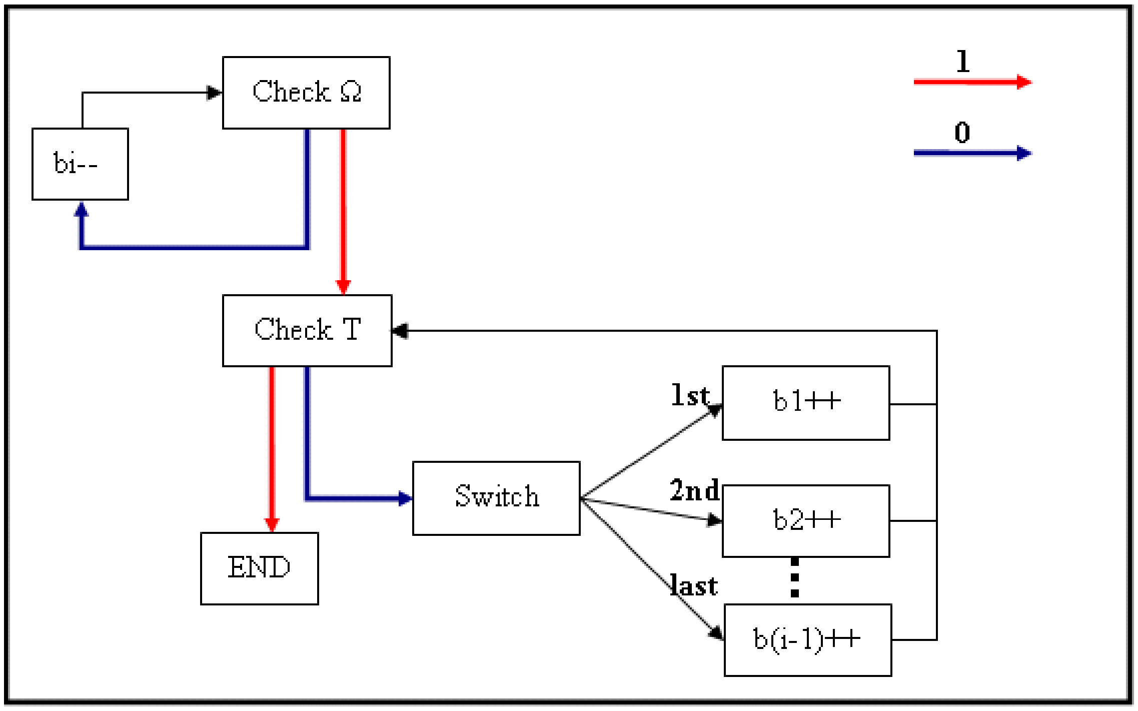

Figure 4.2: Flow diagram of balance_b.m

The first part of the algorithm searches for satisfying omega constraint in the target node. If it is not satisfied, a reduction in one unit of modulation parameter is performed. The reduction implies an increment on transmission time. The minimum value of modulation parameter is stated to two. If it is lower than two an error message is generated and all the output values are equal to -1.

The second part of the algorithm is, once omega constraint is satisfied, to achieve latency constraint. It is managed increasing in one unit the modulation parameter of one different node each time the loop is performed. The higher remaining battery nodes should be raised first.

Finally, infinite execution is a trouble that must be overcome. It can happen if we try to reach the latency constraint increasing the modulation parameter of the balancing group nodes. If we increase modulation parameter when it is already high enough, the latency savings become to be extremely short, and the algorithm falls into infinite execution. In order to avoid this situation, a maximal modulation parameter ($b_{\max}$) is defined. Once it is overcome an error message is generated and all output parameters are equal to infinite.

4.3 The main file

[node,groups_balanced,not_balanced]=main(C,D,Lat,R,s,node_energy)

Several ideas were considered before achieving *main.m*. One of them was to perform modulation parameter in couples, choosing higher battery level nodes with lower ones. However previous theoretical studies led us to the conclusion that weak energy savings can be achieved in target node if we are not willing to spend much energy in balance node. Therefore, more than one balance node should be selected to carry out energy savings in target node. Thus, it was considered the idea of taking groups of nodes depending on the energy battery levels of the target nodes. For instance, a target node with 40% of its total battery would be “helped” by two balancing nodes. A target node with $20\% - 40\%$ battery would take three balancing nodes, and when battery is under 20% the node would be stated as critical target node and four (or even five) balancing nodes would be chosen. The drawback is that is really difficult to state how many nodes have to be assigned depending on the battery level. Thus, another way to perform the balance of the modulation parameters was conceived. In the following, it will be explained the final algorithm, *main.m*, which performs the energy balancing in a whole path of nodes.

This algorithm is on the top level. It manages the previous explained functions. Its input parameters are:

- C,D are vectors that contain the power parameters of a whole path of nodes.
- Lat is the latency of the whole path.
- R is the symbol rate.
- s is the packet size.
- *node_ energy* is a vector that contains the battery level of each node of the path.

The aim of the algorithm is to manage the balancing of energies dynamically; that is, select as many nodes as needed while we have constraints not satisfied.

4.3.1 Node Structure

To begin with, *main.m* uses a structure called node. It is composed of several fields:

node.index index of the node in the path.

node.C transmission power parameter.

node.D electronic power parameter.

node.b current modulation parameter.

node.b0 theoretical optimal modulation parameter.

node.bop theoretical optimal ceiled mod parameter.

node.T transmission time used for one packet of size s.

node.energy remaining battery level.

node.Eused energy used to transmit a packet of size s

node.ratio relationship between $E_{\text{balance}}/E_{\text{opt}}$ where E_{balance} is the energy spent when *balance_ b.m* is performed and E_{opt} is the energy spent when *optimal_ energy.m* is computed.

node.omega factor that tells us the amount of energy that we want to save if the node is selected as target node. Between 0 and 1.

node.gamma factor that tells us the amount of energy that we want to spend if the node is selected as balance node. Greater than 1.

4.3.2 General Description

The diagram flow of the algorithm can be observed in Fig. 4.3.

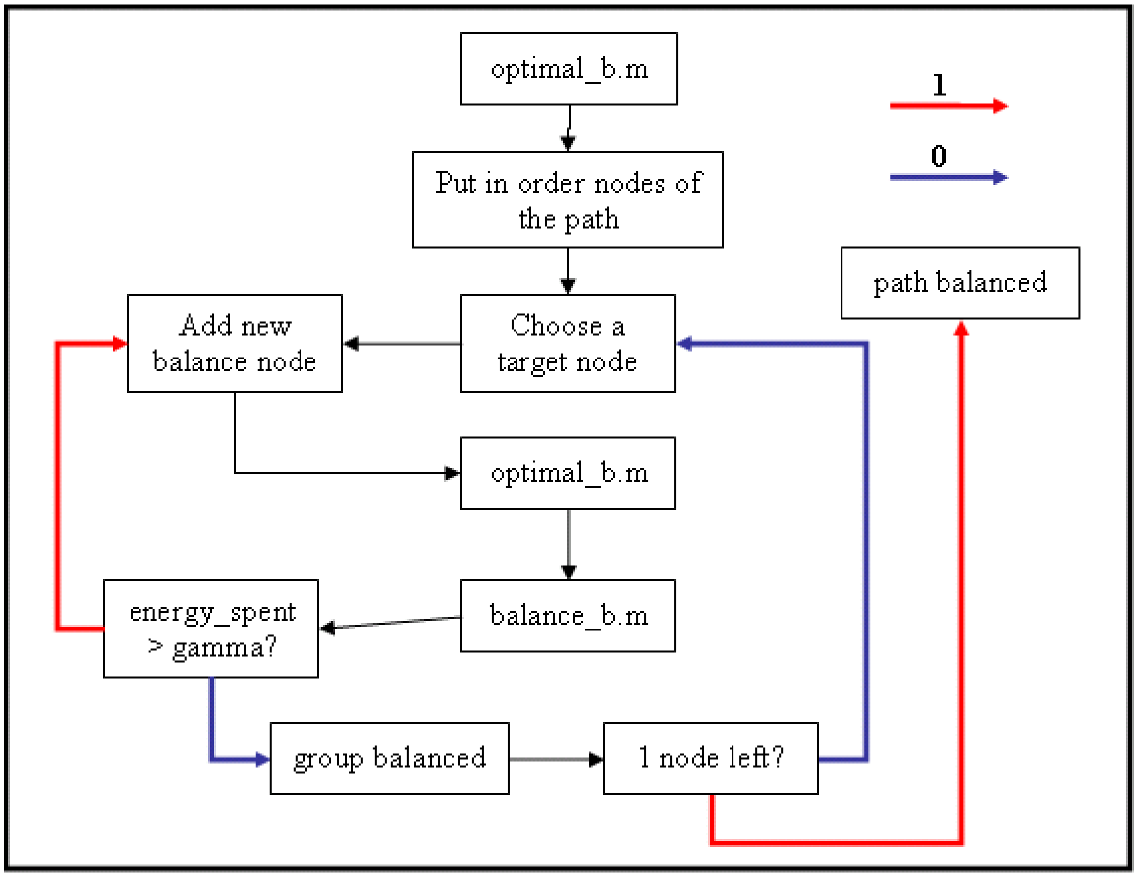

Figure 4.3: Flow fiagram of main.m

Firstly, the optimal modulation node parameters of the whole path are computed together in order to know the theoretical optimal transmission time of each node (*optimal_b.m*). Secondly, we put in order the nodes from lower values of battery level to higher ones. Thus it is easier to manage groups in order to perform the balance of energy. Our strategy is to take one node from the beginning of the list as a target, and then start to take nodes from the end of the list as balancing nodes. Thirdly, we go into a loop where a target is selected and several balancing nodes are assigned to it.

The amount of balancing nodes depends on the following aspect. We need to achieve the stated saving (defined by Ω) in target node. Always a group of nodes is selected before performing the balance of energies. This group is composed of the target node and the balance group of nodes. Hence, *balance_b.m* is executed. It searches a combination of modulation parameters such the target node satisfies Ω. Furthermore, γ has to be satisfied in all nodes that belong to the balance group. Therefore, if the energy spent ratio of some node of balance group overcomes gamma factor we need to add a new node to the group. The higher number of nodes in balance group the lower energy requirements per each node for achieving stated saving in target node. Note that Ω is evaluated inside balance_b.m.

Finally, once both Ω and γ are satisfied, *main.m* continues generating new groups of nodes until it scales all of them.

The outputs of *main.m* are *node*, *groups_balanced*, and *not_balanced*:

- *node* is a vector of structures that contains all the nodes of the given path with the modulation scaling performed.
- *groups_balanced* is a vector that contains the target nodes with their respective balance nodes.
- *not_balanced* is a vector that contains the nodes that have not been balanced.

4.3.3 Details

Energy constraints

First, it has to be mentioned that the decisions taken by the algorithm are based on the file *energyconstraints.m*. Here is where we define the energy constraint for the target node (Ω) and for the balance nodes (γ). An example is shown in the following lines:

```
batlevel=(energy)*100/max_energy;
if batlevel < 5
ome=0.4; gam=1.5;
elseif batlevel > 50
ome=-2; gam=3;
else ome=0.15; gam=2.7;
end
```

Target nodes usually have low levels of battery. Therefore stringent omega parameters have to be determined. This is the reason of having $\Omega = 0.4$ in the range of $0\% - 5\%$ of battery level. $\Omega = 0.4$ means that a 40% of energy will be saved if we are dealing with a target node. On the other hand, balancing nodes use to have high battery levels, and therefore they can spend more energy in order to help target node. This fact is the reason of having $\gamma = 3$ in the range $50\% - 100\%$ of battery level. Finally note that in this last range, Ω is set to a negative value. This means that we allow not saving energy because it does not have any sense to try to save power at these levels of battery in target nodes. Our algorithm assigns to this case the optimal modulation parameter. These constraints are called dynamic constraints due to the fact they depend on the battery level of the node. As we will see in the simulations, static constraints work better for our purposes.

While statement in the code

As it can be observed in the appendix A, the main part of the algorithm is the while statement. Inside it we have five possible situations:

- All constraints have been satisfied, so we update values.
- While and after performing balance_b.m, we have 4 possible situations:
  1. The balance is successful. After the balance, the gamma constraint is checked. If it is satisfied proper values of each node are updated. If it is not satisfied a new balance node is added to the group.
  2. The balance falls in an infinite execution and it stops. The output parameters are assigned to infinite. This means that it is impossible to satisfy the latency constraint with the actual number of balancing nodes. Therefore, the while loop is started again and a new node is added to the balance group.
  3. The balance reaches a modulation parameter equal to one. The output parameters are assigned to -1. This situation means that it is impossible to achieve omega savings in the target node. Therefore, the balance nodes are released and the target node is updated to the optimal value computed at the beginning of the algorithm.
  4. In the above situations described, when a new balance node is added there is the possibility that no more nodes will be available. If we are in this situation, then all the nodes will be updated to the optimal values computed inside the while statement.

For more details, we recommend to read carefully the main.m file attached in the appendix A. Each step performed inside it is extensively commented.

Reorganization of the parameters

On the other hand, in order to run easier *balance_b.m* and *optimal_b.m* we had to reorganize the parameters C and D. It is important to know that the order of the parameters of C and D just before performing *balance_b.m* and *optimal_b.m* were:

$$C\,(1) = C_{\text{target}};\ C\,(2) = C_{\text{balanceNode1}};\ C\,(3) = C_{\text{balanceNode2}};\ldots$$

Where the battery levels were:

$$\zeta_{\text{balanceNode1}} > \zeta_{\text{balanceNode2}} > \ldots > \zeta_{\text{target}} >$$

Our objective was to obtain parameters organized like:

$$C\,(1) = C_{\text{balanceNode1}};\ C\,(2) = C_{\text{balanceNode2}};\ C\,(3) = C_{\text{balanceNode2}};\ldots;C\,(n) = C_{\text{target}}$$

This allowed us to perform the balance in an easier way. After having performed the balance, all the parameters like C,D, and the outputs generated are always restored to the original order.

The blocking node problem

Finally, lots of simulations were performed. Sometimes we found some problems with the balance group node formation. We realised that there were cases where a balancing node blocked the creation of a finite group of balancing nodes, due to the fact it always violated the gamma parameter although new nodes were added. An example based on a simulation is illustrated in the following.

We have a path with 100 nodes. In a certain moment, the algorithm selects the node n 36 as a target node. Node 60,59,... become to be selected as balancing nodes:

Step 1: 2 nodes

Number of node	36	60
Modulation parameter	5	10
Energy ratio	0.5149	5.4577
γ	-	3

Table 4.1: Step 1

Step 2: 3 nodes

Number of node	36	60	59
Modulation parameter	5	9	8
Energy ratio	0.5149	3.0369	1.7372
γ	-	3	3

Table 4.2: Step 2

Step 3: 4 nodes

Number of node	36	60	59	58
Modulation parameter	5	9	8	7
Energy ratio	0.5149	3.0369	1.7372	0.9956
γ	-	3	3	3

Table 4.3: Step 3

...

Step 6: 7 nodes

Number of node	36	60	59	58	57	56	55
Modulation parameter	5	9	8	7	7	7	6
Energy ratio	0.5149	3.0369	1.7372	0.9956	1.0936	1.0869	0.8451
γ	-	3	3	3	3	3	3

Table 4.4: Step 6

Until 14 nodes it was impossible to reach a solution due to the energy ratio (energy used with scaling/energy used without scaling) of node 60 was always greater than the gamma factor. If node 60 had had same properties as node 59, a solution with 3 or 4 nodes would have been achieved.

In order to avoid situations similar to the last one a solution was stated. We defined a new parameter called *max_ group* which determined the maximum possible number of nodes in the balance group. If the algorithm reaches that limit and does not achieve a solution, the *max_ group* nodes plus the target node are restored to their optimal ceiled values and the balance of energies is cancelled for that group of nodes. Mention that the default value for *max_ group* is 5.

With this last explanation we have described how works our approach. We encourage the reader to read carefully the code attached in appendix A.

Chapter 5

Performance Analysis

In this chapter we evaluate the characteristics of the designed algorithm. It is analysed in terms of different topologies, such as theoretical paths, aggregation trees or dense multihop networks. The performance is evaluated with our own simulator, where only some aspects of networks were taken into account. First we carry out an overview of current simulators and we explain how is made our own simulator. Then, for each topology an analysis is performed and the results are shown. Each analysis is based on evaluating the performance with static and dynamic constraints for three different methods: our approach, the algorithm of *Yu and Prasanna* [17] (see page 29) and a non-power algorithm.

5.1 Network Simulator

Presently many simulation and emulation environments for ad-hoc WSN are available. Some examples are:

- The ns-2 is probably the most widely used and most powerful free network simulator, now supporting sensor networks.
- TOSSIM, created for mote-based sensor networks runs the code written for the physical devices. EmStar provides mixed execution environments containing distributed simulators and real sensor nodes.
- GloMoSim, built on PARSEC, provides a scalable environment for wireless networks and provides a wide variety of models for sensor networks as well.
- Qualnet, a fast, scalable, extensible simulation framework, is built on top of a commercially available network simulator, providing high quality networking and sensor stacks.
- SENSE, designed specifically for sensor networks is a fully extensible simulator tool with several provided components and simulation engines.
- PROWLER, a probabilistic network simulator is developed with various radio models and a CSMA MAC layer. RMASE, an application built on the top of PROWLER, provides a layered routing architecture with routing scenario specifications and performance metrics for routing algorithm evaluations.

This partial list of the available simulators shows that there is no ultimate solution; the right choice depends on the purpose of the user and application area.

The ability of developing efficient protocols in all of these simulators requires more effort than this project allowed. RMASE, QUALNET and GLOMOSIM were considered in this Thesis, but after trying to understand their structure, we assumed it was better to start without any simulator and make a new simplified one. This decision was taken due to inexperience in programming protocols for networks, no prior experience with network simulators and the amount of time required to understand and use such simulators.

The simulator was developed in Matlab. The simplifications and assumptions were:

- Each node has its own power characteristics, defined by C and D parameters; its own battery level and other parameters such as latency and modulation.
- Idle listening is not taken into account. This means that the lifetime is measured with the number of transmissions performed by a node. This allows independence from implementation of upper layers as for instance MAC layer.
- The transmission power is equal to the reception power in a node.
- The interference model is not considered. Also an ideal link quality is assumed. This means a 100% packet reception rate.
- The packets sent by nodes have size s.
- The symbol rate R_s for all nodes is the same.
- Due to the fact that lifetime is measured as a number of transmissions, we assumed that sensing nodes take measures from their environment in a synchronized way, and then they forward the data at same times.

The aim of our simulator is to observe if the lifetime is increased or not. For carrying it out, the idea is to count how many transmissions a node is able to perform. Depending on the kind of topology, this count is performed in a different way:

- In a theoretical path of nodes, there is only one node that is the source and another one that is the sink. Therefore, when a packet is sent, all sensors perform one transmission and one reception. Our simulation file performs this in an infinite loop. When one of the nodes dies because it has no more energy for transmitting, it generates an error message, the simulation stops and presents the total number of transmissions performed in the path.
- For the rest of the topologies, multiple paths are considered to be crossing the same node. We have to remember that we assumed that all the sources transmitted packets in a synchronized way. All of them send data at the same times. Therefore, if x sources send a packet at time t, these packets will take several paths in the network. Our simulator studies what happens in one of several paths of network. Now, nodes can transmit more than one packet each time due to the fact that they can be part of several paths. Thus, as our algorithm is applied to all paths, the results obtained for one path can be extrapolated to other ones and therefore, we can

say that the results for one path are the same for all paths of the whole network. This can be considered a good approximation if we assume that the traffic load of the network is distributed equally through all paths of the network. Our simulation models for this kind of network define a vector that scales the number of transmissions and receptions that nodes perform depending on how many paths crosses them. Usually the nodes closer to the sink/s have higher scaling factors because they perform the transmission of the packets coming from several sources. However, sensing nodes as they only perform one transmission; their scaling factor is equal to one. For each topology, a specific explanation is given in order to understand how the simulations are performed. The corresponding files of the simulators for each topology can be found in the appendix B.

Finally note that in each simulation, three different algorithms are tested. The first one is our approach, which is called *Balancing* method. The second one is the approach of Yu and Prasanna [17], which is called *Power-Aware* protocol. The last one is a mechanism that transmits with the maximum modulation parameter (equal to 10), which we call *Non-Power-Aware* method.

5.2 Theoretical Path

As we mentioned, our work is based on the study performed by Yu and Prasanna [17]. They evaluated their algorithm in a simple path of nodes and this is the reason that we started our analysis with this structure (see Fig. 5.1).

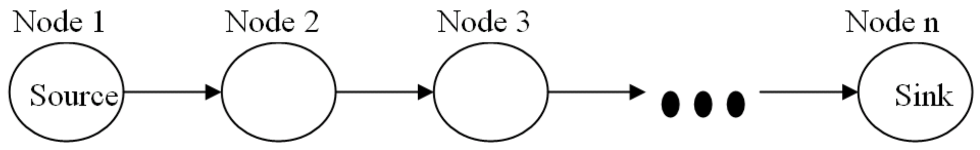

Figure 5.1: Simple path of nodes

Although our algorithm is not designed for this kind of path, several simulations were performed in order to have a better understanding of how works our algorithm. Thus, a first simulation with static constraints was carried out. Remember that Ω specifies the saving that we want to obtain in the target node, and γ the over-consumption in the balancing nodes. The parameters for this simulation were:

- $\Omega = 0.4$ and $\gamma = 2.7$ (static constraints)
- 10 nodes
- $s = 100$ bits; $R = 1$ MHz and $T = 15 \cdot 10^{-5}$ s.
- $C = \left[0.8 \cdot 10^{-7}, 1.2 \cdot 10^{-7}\right]$ (uniform distribution)

- $D = \left[1.8 \cdot 10^{-7}, 2.2 \cdot 10^{-7}\right]$ (uniform distribution)
- Maximum battery level $\zeta_{max} = 10$ Joules
- Uniform distribution for battery levels $\zeta = [0, 10]$ Joules

The battery levels of each node are the following ones:

Node index	1	2	3	4	5	6	7	8	9	10
ζ	1.1375	7.7211	5.0059	9.3863	5.1939	4.7678	8.616	5.7264	2.8238	7.0492

Table 5.1: Battery levels for each node of the path

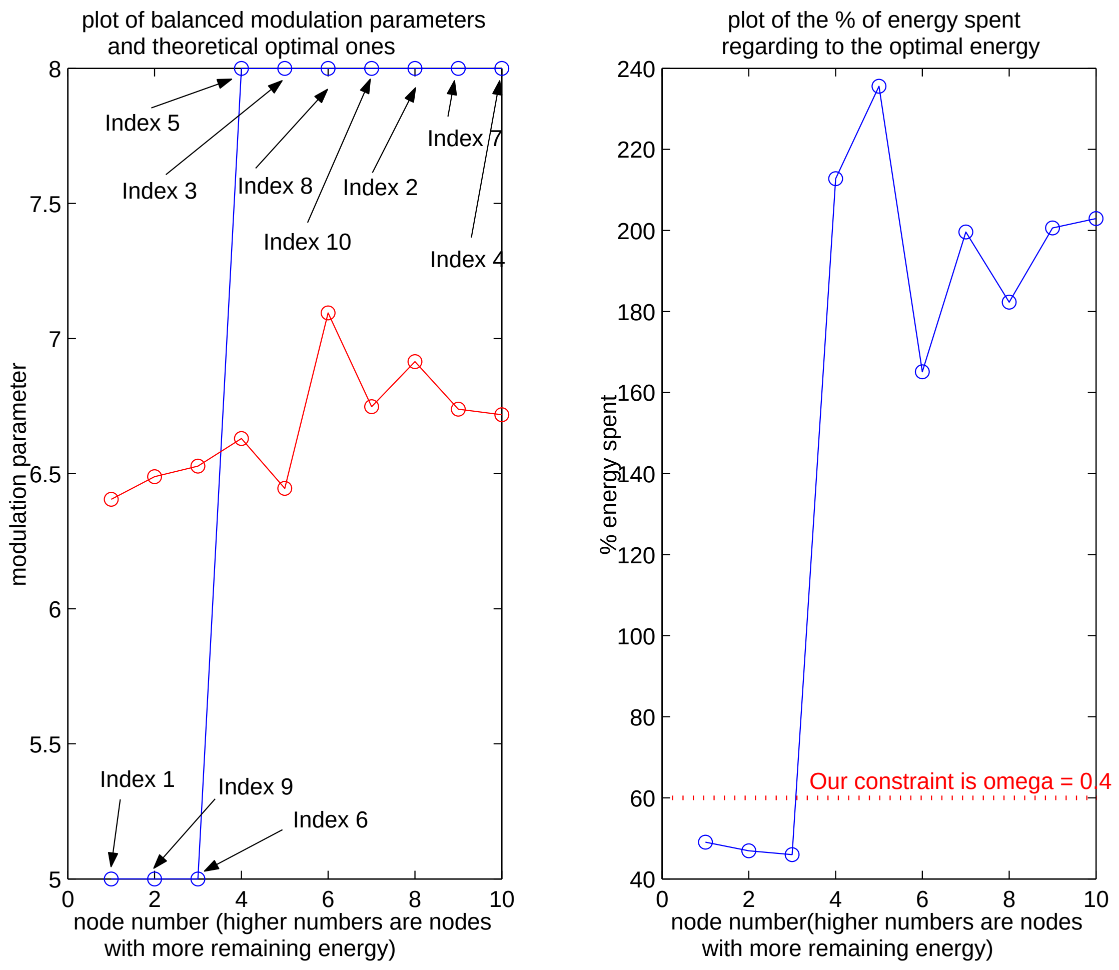

Figure 5.2: Behaviour of our method

Fig. 5.2 shows the behaviour of our algorithm at the first iteration. The nodes are represented in order, from lower battery to higher one. In the plot on the left, we have in blue the scaled modulation parameters, and in red the theoretical optimal ones. Note that the theoretical modulation parameters are not integers. In the plot on the right, the percentage of energy spent regarding the optimal energy is presented for each node. If we take a look on both plots, it can be observed how the nodes with less energy (1.1375 J, 2.8238 J, and 4.7678 J) are chosen as target nodes and around 50% of energy is saved, while in the rest of nodes, around 200% of energy (regarding the optimal) is spent. Furthermore, we can know the groups formed if we look at the output of the simulation. Nodes 10 (index 4) and 9 (index 7) were selected for saving energy in node 1 (index 1). Nodes 8 (index 2), 7 (index 10), 6 (index 8) were selected for saving energy in node 2 (index 9). And nodes 5 (index 3), 4 (index 5) were selected for saving energy in node 3 (index 6). The modulation parameters were balanced satisfying ($T = 14.75 \cdot 10^{-5}$ s) the latency constraint ($T = 15 \cdot 10^{-5}$ s).

The results regarding the lifetime were the following ones:

Lifetime	Balancing	Power-Aware	Non-Power-Aware
Nº of transmissions	14724	8935	941

Table 5.2: First experiment results

More simulations with different uniform distributions for node battery levels were performed. Some results are presented here:

Lifetime	Balancing	Power-Aware	Non-Power-Aware
Nº of transmissions $\zeta = [0.5, 10]$	6473	3918	414
Nº of transmissions $\zeta = [1, 10]$	12945	7855	827
Nº of transmissions $\zeta = [1.5, 10]$	19417	11782	1241
Nº of transmissions $\zeta = [2, 10]$	23456	15709	1654
Nº of transmissions $\zeta = [5, 10]$	30778	23458	4212
Nº of transmissions $\zeta = [7, 10]$	37627	36406	6549
Nº of transmissions $\zeta = [10, 10]$	44550	49200	8270

Table 5.3: First experiment results II

Table 5.3 shows that as nodes have more energy, our algorithm is less efficient. However, when there are big energy differences between nodes, our method yields a better performance. Our algorithm is designed for solving this kind of differences, which are given in nodes that form part of multiple paths and not only one. This is the reason that it has a worse performance for nodes with more energy.

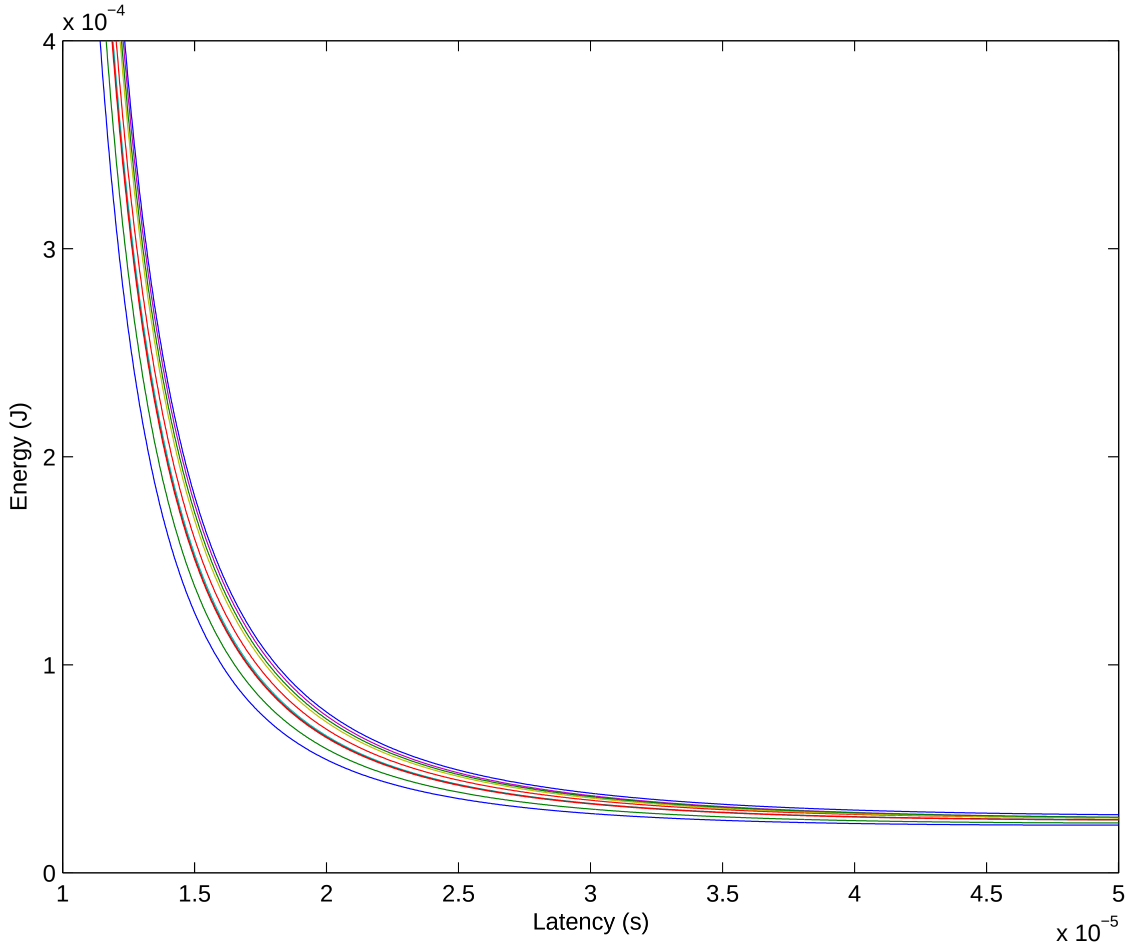

Figure 5.3: Energy curves

On the other hand, latency was changed to many different values. We observed that for high values of latency, savings were more difficult to achieve since there is more separation (in terms of latency) between modulation parameters, and the energy curves are flatter in this region (See Fig. 5.3). Therefore, depending on the latency, different energy constraints (Ω and γ) should be selected. From many simulations we reach the conclusion that:

- Nodes with average latency in the range $\left[2 \cdot 10^{-5}, 3.5 \cdot 10^{-5}\right]$ should have $\Omega = 0.2$ and $\gamma = 3$.
- Nodes with average latency in the range $\left[0.5 \cdot 10^{-5}, 2 \cdot 10^{-5}\right]$ should have Ω between 0.4 and 0.6 and $\gamma = 3$.

Besides these static constraints, dynamic constraints were evaluated as well. But now, instead of creating a path of 10 nodes, we created a path with 100 nodes. The parameters were the same ones, with the difference in the energy constraints. Hence, we used the following constraints:

```
if batlevel < 10
ome=0.4; gam=1;
elseif batlevel > 50
ome=-2; gam=3;
else
ome=0.15; gam=2.7;
end
```

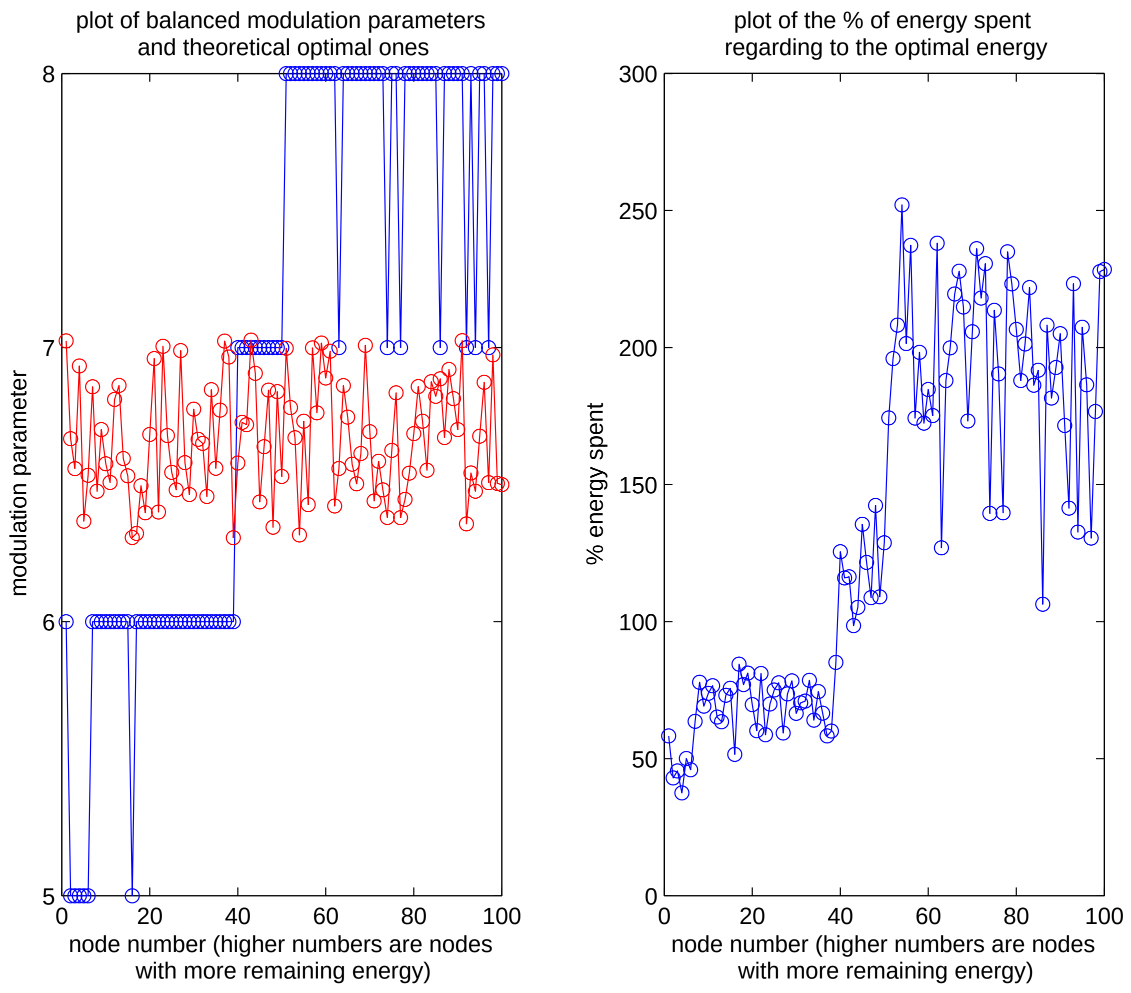

Figure 5.4: Behaviour for dynamic constraints

Fig. 5.4 shows how our algorithm manages the nodes when dynamic parameters are used. To our surprise, regarding lifetime the results were similar or even worse. With the simulations of other topologies (which will be showed in the following sections) we reached the conclusion that it is better to use static constraints than run our algorithm with dynamic ones. The reason is that dynamic constraints are more conservative than static ones. It can be observed in Fig. 5.4 that there are three levels of modulation parameters applied to the nodes. The nodes with lower modulations (around 40 nodes) are the target nodes, which are saving around 40% energy. Then we have nodes with higher modulations (around 50). They are balancing nodes and they are spending around 200% of energy. Finally, another group of nodes is in the middle (around 10). They are nodes that have the rounded optimal modulation values. From the point of view of nodes that only belong to one path, this is optimal. However, from the point of view of nodes that belong to several paths it is not optimal. Therefore, more remaining energy (it is not going to be used anymore) when the path has died is not used for helping target nodes.

5.3 Tree-based Topology

Many papers [12, 16, 22, 20, 21] describing WSN present the traffic pattern many-to-one, that is, they represent a sort of tree topology. These kinds of topologies are appropriate for testing the performance of our algorithm since they present a non-uniform distribution of the energy consumed along a path. Nodes closer to the sink (the main node) consume much more energy than nodes closer to the lowest levels of the tree. In this section, two different trees are simulated. The first is a spanning tree with data fusion. The other is a random tree without any symmetry.

5.3.1 Data Aggregation Tree

A spanning tree composed of 31 nodes and 5 levels was considered to test the protocol (Fig. 5.5).

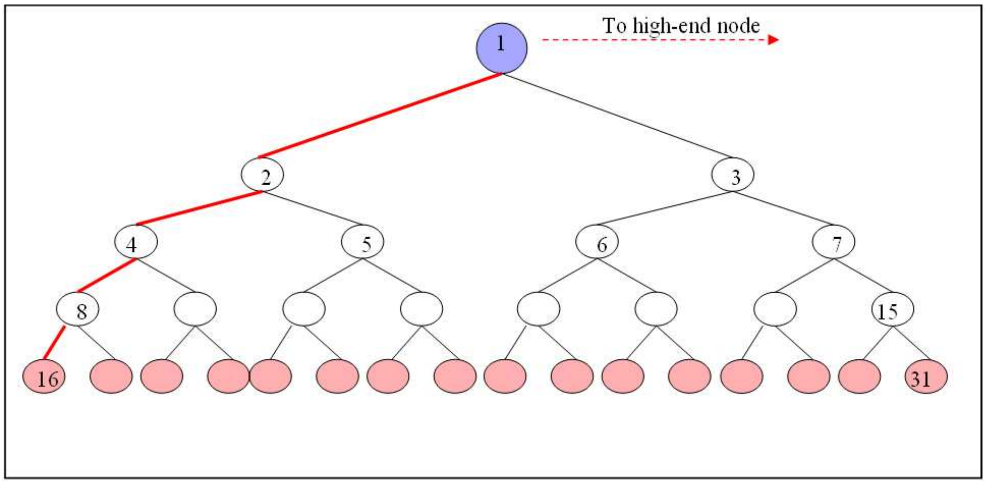

Figure 5.5: Spanning tree

Usually this kind of tree uses data aggregation or data fusion. As we commented on Section 2.1.5, this means that each node aggregates and gathers information from all its children and routes the packets to the sink. Due to the level of correlation between the information a reduced size packet is generated. It is dependent of the correlation level between information, k, that is in the range [0, 1];

and the number of source nodes in the subtree routed, d. The relationship between the new packet generated s' and the previous one s can be abstracted [22] using the following equation:

$$s' = \frac{ds}{dk - k + 1} \tag{5.1}$$

For our analysis we also assumed the following abstraction:

- First, all sensor nodes (16,17,18,19,...,31) transmit packets at same times with a latency constraint.
- Secondly, if $k = 0$ each node will transmit two times more the amount of packets than its children do. This means that two times more energy will be spent. Therefore, if $k \neq 0$ we will have a factor ($^{s'}/_{s}$) that will determine the amount of energy spent for each packet transmission.

The aim of this abstraction it to try to emulate the real node consumption of a whole network when nodes receive packets form multiple sources, due to the fact that in previous simulations it was considered that each node only transmitted one packet from one source and not from multiple ones.

Therefore, the simulation of one path of the spanning tree was performed. We chose the path composed of nodes 1, 2, 4, 8 and 16. Computing equation 5.1 we had that the energy consumption factors (Z_i) were (for $k = 0.5$):

Node	16	8	4	2	1
Z	1	1.33	1.78	2.37	3.16

Table 5.4: Energy consumption distribution along the path

For instance, node 2 with a factor of 2.37 means that when all the sensor nodes transmit one packet at the same time, node 2 will spend $E \cdot 2.37$ Julies due to the correlation between the packets sent from different sources.

The parameters for the simulation were the following ones:

- Ω = variable and $\gamma = 3$ (static constraints)
- For dynamic constraints see energyconstraints.m in the appendix A
- 5 nodes
- $s = 100$ bits; $R = 1$ MHz and T = variable
- $C = \left[0.8 \cdot 10^{-7}, 1.2 \cdot 10^{-7}\right]$ (uniform distribution)
- $D = \left[1.8 \cdot 10^{-7}, 2.2 \cdot 10^{-7}\right]$ (uniform distribution)
- Maximum battery level $\zeta_{max} = 10$ Joules

At this point in the simulations, we did not know wether it was optimal to run the algorithm from the beginning of the path, or in contrast, it was efficient to use optimal modulation values until the energy level of some node became to be lower than certain level. Therefore, Power-aware protocol was mixed with our Balancing algorithm. When one of the nodes achieves a specific starting level, it starts out with the balancing protocol. If its energy is above that level, the Power-Aware protocol is run from the start. The Tables 5.5 and 5.10 show the increase in lifetime regarding the starting level and the different energy constraints.

Lifetime ($T = 7.5 \cdot 10^{-5}$s)	Balancing (static: $\Omega = 0.6$)	Balancing (static: $\Omega = 0.4$)	Balancing (dynamic)
Nº of transmissions starting level = 10%	20130	20339	20339
Nº of transmissions starting level = 20%	20725	21181	21410
Nº of transmissions starting level = 30%	21282	22047	22668
Nº of transmissions starting level = 40%	21834	22893	23558
Nº of transmissions starting level = 50%	22372	23738	24286
Nº of transmissions starting level = 60%	22876	24590	24415
Nº of transmissions starting level = 70%	23363	**25443**	24544
Nº of transmissions starting level = 80%	23451	25402	24609
Nº of transmissions starting level = 90%	23287	25389	24697
Nº of transmissions starting level = 100%	22890	25330	24826

Table 5.5: Nº of transmissions for $T = 7.5 \cdot 10^{-5}$s.

The total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 18013$$

And for the Non-aware protocol:

$$N_{tx} = 3245$$

It can be observed that the best results are for the static constraints with omega equal to 0.4. The best life-time is for a starting level of 70%. However the difference to the other starting levels is negligible. The **increase of the lifetime** regarding the Power-aware protocol is approximately 41% **higher** (lifetime is 1.41 times higher).

Lifetime ($T = 7.5 \cdot 10^{-5}$s)	Balancing (static: $\Omega = 0.4$)	Balancing (static: $\Omega = 0.2$)	Balancing (dynamic)
N$^{\underline{o}}$ of transmissions starting level = 10%	77043	81962	79011
N$^{\underline{o}}$ of transmissions starting level = 20%	77043	86880	82096
N$^{\underline{o}}$ of transmissions starting level = 30%	77043	89211	85219
N$^{\underline{o}}$ of transmissions starting level = 40%	77043	91172	88331
N$^{\underline{o}}$ of transmissions starting level = 50%	77043	93125	91442
N$^{\underline{o}}$ of transmissions starting level = 60%	77043	95077	92319
N$^{\underline{o}}$ of transmissions starting level = 70%	77043	97036	93224
N$^{\underline{o}}$ of transmissions starting level = 80%	77043	99004	94168
N$^{\underline{o}}$ of transmissions starting level = 90%	77043	100972	95045
N$^{\underline{o}}$ of transmissions starting level = 100%	77043	**102939**	95094

Table 5.6: N$^{\underline{o}}$ of transmissions for $T = 15 \cdot 10^{-5}$s.

The total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 77043$$

And for the Non-aware protocol:

$$N_{tx} = 3245$$

The best results for a starting level equal to 100% was achieved for a static parameter equal to 0.2. The **increase of the lifetime** regarding the Power-aware protocol is approximately 34% **higher** (1.34 times higher).

From the last two tables, we can state that the proper starting level is when all the nodes have 100% of its battery level. Also, we can observe that there is a reduction on the increase of the lifetime due to the different latency used. Therefore, different simulations were performed in order to visualize the dependence of the savings with the **total latency of the path** T. Table 5.7 shows the increase in the lifetime regarding the Power-aware protocol for several values of latency. These values were selected in the range of a modulation parameter between $b = 3$ and $b = 9$. The plot of all of these values together can be seen in Fig. 5.6.

$\Omega = 0.4$			$\Omega = 0.2$			*dynamic*		
b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)
9	*5.55*	124	*9*	*5.55*	118	*9*	*5.55*	102
8.5	*5.90*	59	*8.5*	*5.90*	57	*8.5*	*5.90*	58
8	*6.25*	25	*8*	*6.25*	22	*8*	*6.25*	12
7.5	*6.65*	58	*7.5*	*6.65*	58	*7.5*	*6.65*	54
7	*7.10*	110	*7*	*7.10*	95	*7*	*7.10*	90
6.5	*7.70*	52	*6.5*	*7.70*	51	*6.5*	*7.70*	50
6	*8.35*	24	*6*	*8.35*	16	*6*	*8.35*	20
5.5	*9.10*	52	*5.5*	*9.10*	34	*5.5*	*9.10*	45
5	*10.0*	26	*5*	*10.0*	25	*5*	*10.0*	16
4.5	*11.1*	37	*4.5*	*11.1*	35	*4.5*	*11.1*	36
4	*12.5*	4	*4*	*12.5*	7	*4*	*12.5*	14
3.5	*14.3*	0	*3.5*	*14.3*	30	*3.5*	*14.3*	14
3	*16.65*	0	*3*	*16.65*	62	*3*	*16.65*	42

Table 5.7: Analysis with variable latency

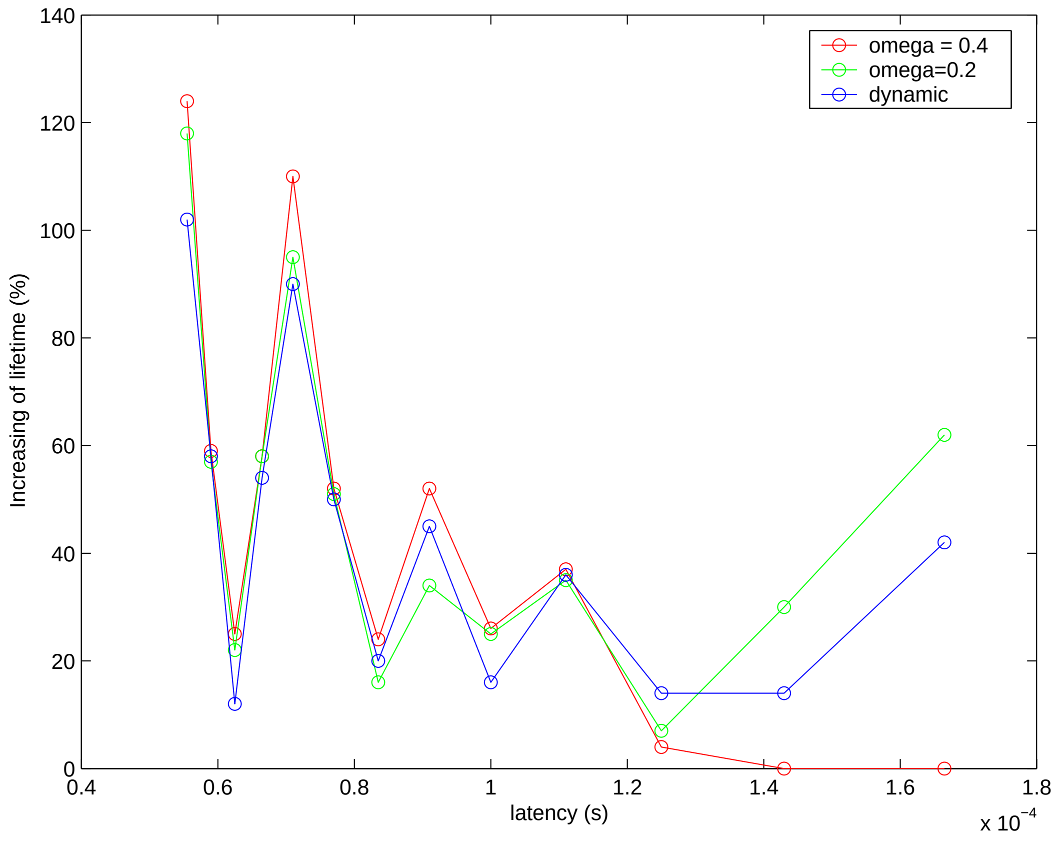

Figure 5.6: Performance for different latency constraints

The best choice has a strong dependence on the application being used. For instance, applications that work with high modulation parameters (low latency) can have a favourable increase in their lifetime. However, applications with low modulations (low energy consumption), have smaller increases in their lifetime when our algorithm is applied. Therefore, one way of measure the best option, is to assume that all the latencies analysed, have the same probability of being used. Thus, by computing the mean value of every lifetime increments we can obtain the best choice or combination of choices. Thus,

- mean $(\Omega = 0.2) = 46.9692\%$
- mean $(\Omega = 0.4) = 44.1369\%$
- mean $(\mathit{dynamic}) = 42.4862\%$

If we use a combination of omegas we obtain the best choice:

- omega $= 0.4 \in T = \left[5.55 \cdot 10^{-5}, 11.1 \cdot 10^{-5}\right]$ s.
- omega $= 0.2 \in T = \left[11.1 \cdot 10^{-5}, 16.65 \cdot 10^{-5}\right]$ s.

Then we obtain:

- mean $(\Omega_{\mathrm{MIXED}}) = 51.3646\%$ (1.51 times higher)

5.3.2 Random tree

A different spanning tree topology was simulated as well. Now, the path of our study was a part of a random tree with higher length. Specifically, it was composed of 10 nodes. The topology is represented in Fig. 5.7.

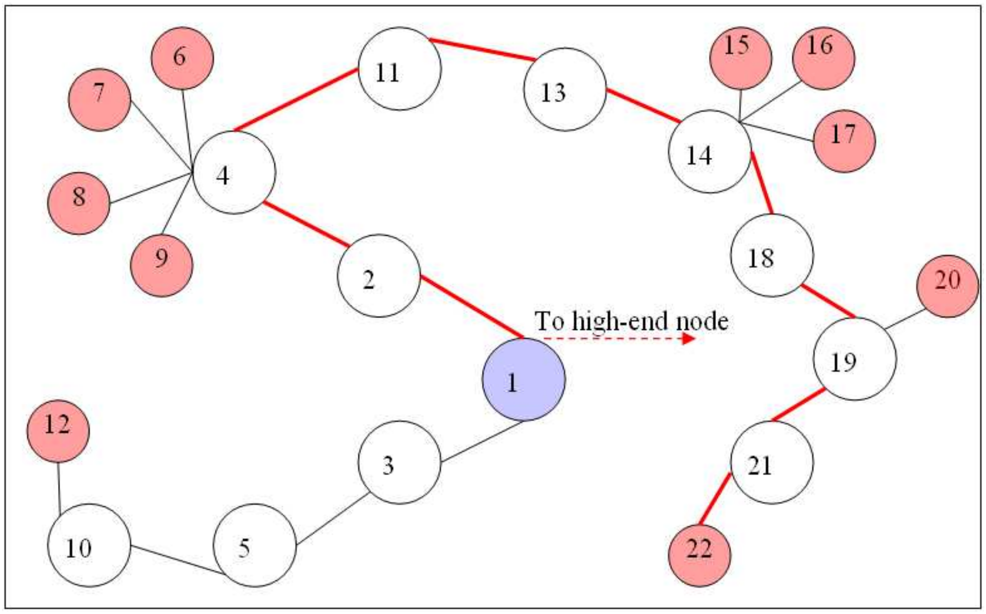

Figure 5.7: Random tree topology

The correlation factor was fixed to $k = 0$ in order to evaluate the worst case. When $k = 0$ no correlation is available. In addition, the differences of energy consumption between different nodes are more relevant and therefore the lifetime for other protocols is even lower.

The performance obtained for this topology with $k = 0$ can be approximately similar to the performance obtained for topologies like multi-hop mesh graphs and for techniques based on data-centric routing which does not use data-gathering with compression or fusion.

The parameters for the simulation were the following ones:

- Ω = variable and $\gamma = 3$ (static constraints)
- For dynamic constraints see energyconstraints.m in the appendix A
- 10 nodes
- $s = 100$ bits; $R = 1$ MHz and T = variable
- $C = \left[0.8 \cdot 10^{-7}, 1.2 \cdot 10^{-7}\right]$ (uniform distribution)
- $D = \left[1.8 \cdot 10^{-7}, 2.2 \cdot 10^{-7}\right]$ (uniform distribution)
- Maximum battery level $\zeta_{max} = 10$ Joules

The distribution of consumptions over the relevant path (the one which is in red colour in Fig. 5.7) was:

Node	22	21	19	18	14	13	11	4	2	1
Z	1	1	2	2	5	5	5	9	9	9

Table 5.8: Energy consumption distribution along the path

As in the last simulation, we tested our algorithm with different starting levels. The results can be observed in Table 5.9[1].

The total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 5793$$

And for the Non-aware protocol:

$$N_{tx} = 1041$$

Again the best results are for the static constraints. It can be observed that for omega equal to 0.4 and a starting level equal to 90%, the **increase** of the life-time regarding the Power-aware protocol is approximately 64% **higher**. The explanation of these results rely on the fact that our protocol tries to balance the energy consumptions, trying to reduce the consumption in those nodes (see Fig. 5.7) with higher energy consumption (like nodes 1, 2 and 4) and increasing the consumption in those nodes with lower energy requirements (like nodes 22, 21, 19 and 18).

[1] the parameter *max_ group* is set to 9 nodes for $\Omega = 0.6$.

Lifetime ($T = 15 \cdot 10^{-5}$ s)	Balancing (static: $\Omega = 0.6$)	Balancing (static: $\Omega = 0.4$)	Balancing (dynamic)
N$^{\underline{o}}$ of transmissions starting level = 10%	7553	6833	6833
N$^{\underline{o}}$ of transmissions starting level = 20%	8043	7873	7548
N$^{\underline{o}}$ of transmissions starting level = 30%	8376	8913	7939
N$^{\underline{o}}$ of transmissions starting level = 40%	8524	9221	8330
N$^{\underline{o}}$ of transmissions starting level = 50%	8748	9296	8721
N$^{\underline{o}}$ of transmissions starting level = 60%	8676	9357	8739
N$^{\underline{o}}$ of transmissions starting level = 70%	8489	9389	8690
N$^{\underline{o}}$ of transmissions starting level = 80%	8471	9209	8707
N$^{\underline{o}}$ of transmissions starting level = 90%	8338	**9525**	8710
N$^{\underline{o}}$ of transmissions starting level = 100%	8169	9495	8726

Table 5.9: N$^{\underline{o}}$ of transmissions for $T = 15 \cdot 10^{-5}$s.

The simulation was performed also for another different latency constraint. The results can be observed in Table. 5.10[2].

The total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 28991$$

And for the Non-aware protocol:

$$N_{tx} = 1041$$

For the new latency constraint, the performance offers an **increase** of the life-time around 37% **higher**.

It can be observed for static constraints, when omega is so high that the number of transmissions is equal to the Power-aware protocol. The reason is that with the new latency constraint, we are working with modulation parameters between 2 and 5. Between these parameters, the energy consumption increase is really small and therefore it is really difficult to reach omega constraint. Thus, our protocol just performs the optimal modulation values and doesn't apply balancing.

Moreover, as we saw before, the yield of our protocol is worse as latency increases. As in the other topology, we made a simulation for each latency, taking the best case found in the last two tables, we mean, we assumed a starting level of 100% of the node battery (Table 5.11).

[2]the parameter *max_ group* is set to 9 nodes for $\Omega = 0.4$.

Lifetime ($T = 30 \cdot 10^{-5}$s)	Balancing (static: $\Omega = 0.4$)	Balancing (static: $\Omega = 0.2$)	Balancing (dynamic)
N^o of transmissions starting level = 10%	28991	30743	30743
N^o of transmissions starting level = 20%	28991	32496	32178
N^o of transmissions starting level = 30%	28991	34249	33294
N^o of transmissions starting level = 40%	28991	36002	34410
N^o of transmissions starting level = 50%	28991	37269	35526
N^o of transmissions starting level = 60%	28991	37914	35524
N^o of transmissions starting level = 70%	28991	38282	35506
N^o of transmissions starting level = 80%	28991	38966	35488
N^o of transmissions starting level = 90%	28991	39214	35530
N^o of transmissions starting level = 100%	28991	**39675**	35512

Table 5.10: N^o of transmissions for $T = 30 \cdot 10^{-5}$s.

$\Omega = 0.4$			$\Omega = 0.2$			*dynamic*		
b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)
9	*11.1*	122	*9*	*11.1*	117	*9*	*11.1*	93
8.5	*11.8*	59	*8.5*	*11.8*	77	*8.5*	*11.8*	56
8	*12.5*	38	*8*	*12.5*	42	*8*	*12.5*	29
7.5	*13.3*	77	*7.5*	*13.3*	106	*7.5*	*13.3*	81
7	*14.2*	104	*7*	*14.2*	92	*7*	*14.2*	89
6.5	*15.4*	83	*6.5*	*15.4*	100	*6.5*	*15.4*	77
6	*16.7*	41	*6*	*16.7*	32	*6*	*16.7*	26
5.5	*18.2*	75	*5.5*	*18.2*	92	*5.5*	*18.2*	62
5	*20.0*	36	*5*	*20.0*	39	*5*	*20.0*	26
4.5	*22.2*	55	*4.5*	*22.2*	79	*4.5*	*22.2*	55
4	*25.5*	0	*4*	*25.5*	42	*4*	*25.5*	15
3.5	*28.6*	0	*3.5*	*28.6*	38	*3.5*	*28.6*	30
3	*33.3*	0	*3*	*33.3*	0	*3*	*33.3*	24

Table 5.11: Analysis for several latencies

If we show all together in a plot we obtain the Fig. 5.8.

As we said in the other topology, one way to know which solution is better is perform the mean:

- mean $(\Omega = 0.2) = 65.6987\%$ (1.66 times higher)
- mean $(\Omega = 0.4) = 53.0730\%$
- mean $(dynamic) = 51.0268\%$

where the case for $\Omega = 0.2$ was the best choice for this topology.

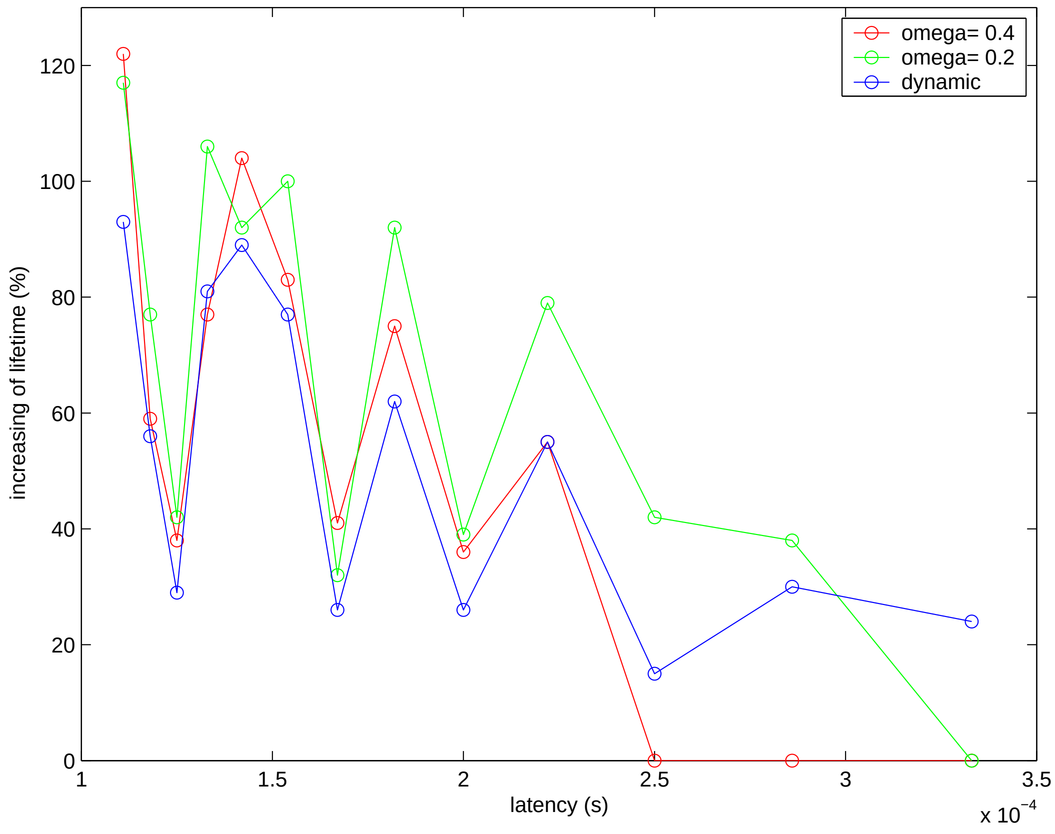

Figure 5.8: Performance for different latency constraints

5.3.3 Conclusions for Tree Topology

From all the results obtained, we can state that the proper starting level is since the beginning of the path, that is, when all the nodes have 100% of its battery level.

The lifetime increase has a decreasing tendency as well as the latency is higher (or modulation decreases).

From both topologies, we obtained different performance. For the first topology, the spanning tree with data fusion, we had an increase of 51.36%. For the second, the random tree, the lifetime increase was 65.7% more. Then, if we remember the energy consumption factors for both topologies:

Node	16	8	4	2	1
Z	1	1.33	1.78	2.37	3.16

Table 5.12: Energy consumption distribution along the path

Node	22	21	19	18	14	13	11	4	2	1
Z	1	1	2	2	5	5	5	9	9	9

Table 5.13: Energy consumption distribution along the path

There are two possible explanations:

- The first one may be the difference in consumption between the node that spends more energy and the node that spends less. Since this point of view, we can state that the higher the consumption differences in a path of nodes, the higher the increase of lifetime of our algorithm.
- The second could be that as the random tree has a longer path, more balance groups are generated and that leads to higher increases of the lifetime.

In order to be able to know if these two conclusions are truthful, we performed simulations with one hundred nodes in the following section.

5.4 Multihop Networks

In the following a large path of sensors will be simulated. Multi-hop dense networks use to sense information and forward it to a sink node which means that nodes closer to the sink carry heavier traffic loads. Therefore the network life-time can be limited by those nodes with heavier data load and therefore greater power consumption.

In order to represent a uniformly distributed multi-hop dense network, an exponential distribution of energy consumptions for the nodes of a path is used. We choose an exponential distribution because we suppose that nodes closer to the sink spend much more energy than sensing nodes.

The parameters for the simulation are:

- Ω = variable and $\gamma = 3$ (static constraints)
- For dynamic constraints see energyconstraints.m in the appendix A
- 100 nodes
- $s = 100$ bits; $R = 1$ MHz and T = variable
- $C = \left[0.8 \cdot 10^{-7}, 1.2 \cdot 10^{-7}\right]$ (uniform distribution)
- $D = \left[1.8 \cdot 10^{-7}, 2.2 \cdot 10^{-7}\right]$ (uniform distribution)

- Maximum battery level $\zeta_{max} = 10$ Joules

Data fusion is not considered. Therefore the consumption factors of each node are integers. Two different exponential distributions were evaluated and are presented here.

5.4.1 Exponential Distribution 1

The power consumption distribution along the path is showed in the Fig. 5.9.

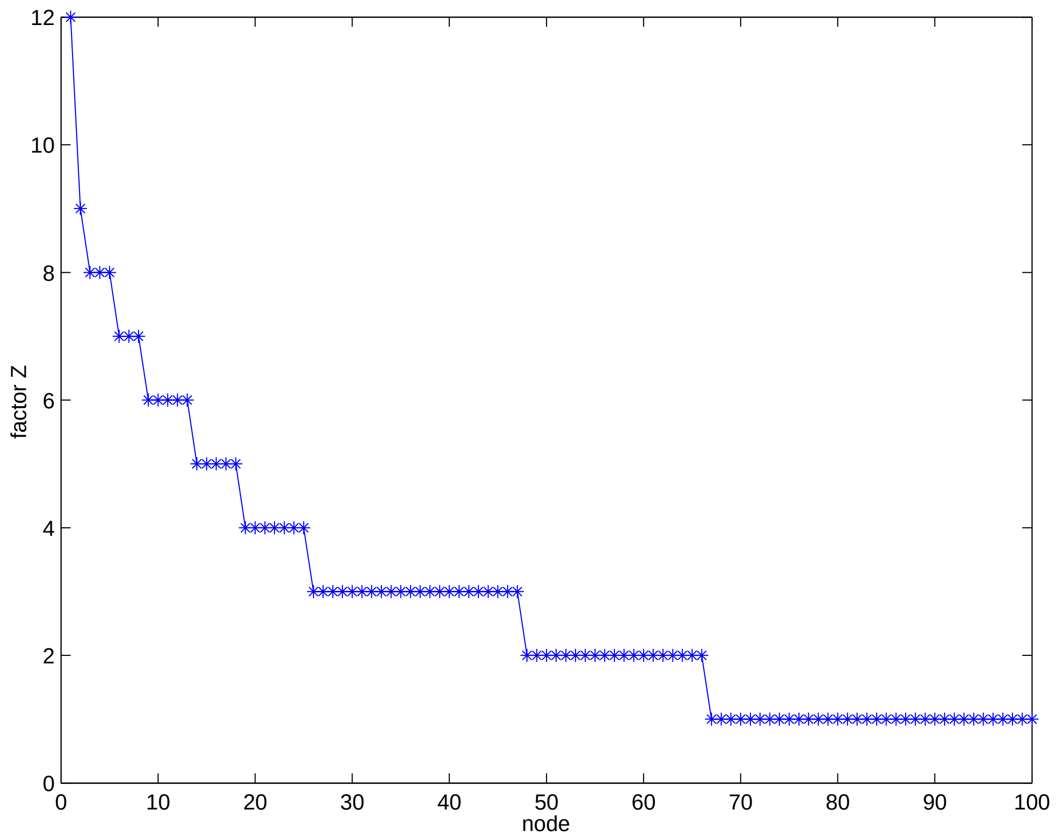

Figure 5.9: Exponential consumption along the path of nodes

The simulation was performed as in the other topologies. First two latencies were evaluated for have an idea if the best choice is to start running our algorithm since the beginning or not. Then an analysis for the best cases was carried out for different latencies.

Lifetime ($T = 150 \cdot 10^{-5}$s)	Balancing (static: $\Omega = 0.6$)	Balancing (static: $\Omega = 0.4$)	Balancing (dynamic)
N$^{\underline{o}}$ of transmissions starting level = 10%	6312	5591	5591
N$^{\underline{o}}$ of transmissions starting level = 20%	7877	6436	5818
N$^{\underline{o}}$ of transmissions starting level = 30%	8079	7283	6394
N$^{\underline{o}}$ of transmissions starting level = 40%	9010	7408	5824
N$^{\underline{o}}$ of transmissions starting level = 50%	10578	8254	7171
N$^{\underline{o}}$ of transmissions starting level = 60%	9320	9459	7065
N$^{\underline{o}}$ of transmissions starting level = 70%	10405	9766	6831
N$^{\underline{o}}$ of transmissions starting level = 80%	9607	10971	7072
N$^{\underline{o}}$ of transmissions starting level = 90%	12510	9657	7134
N$^{\underline{o}}$ of transmissions starting level = 100%	**12776**	**12568**	6880

Table 5.14: N$^{\underline{o}}$ of transmissions for $T = 150 \cdot 10^{-5}$s.

The total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 4746$$

And for the Non-aware protocol:

$$N_{tx} = 855$$

From Table 5.14[3] we can see that again the best cases are for a starting level of 100%. However, when latency is high (Table 5.15[4]), we observe a decrease in the yield when we use high omegas (0.4) or dynamic constraints. The starting level is not 100% and also the performance (when $\Omega = 0.4$) can be lower than the Power-aware protocol. This means that a combination of high omegas for low latencies and low omegas for high latencies should be used as we observed in the first experiment.

[3]the parameter *max_ group* is set to 9 nodes for $\Omega = 0.6$.
[4]the parameter *max_ group* is set to 9 nodes for $\Omega = 0.4$.

Lifetime ($T = 300 \cdot 10^{-5}$ s)	Balancing (static: $\Omega = 0.4$)	Balancing (static: $\Omega = 0.2$)	Balancing (dynamic)
N$^{\underline{o}}$ of transmissions starting level = 10%	27800	29257	29257
N$^{\underline{o}}$ of transmissions starting level = 20%	26979	29780	29465
N$^{\underline{o}}$ of transmissions starting level = 30%	26159	30304	29468
N$^{\underline{o}}$ of transmissions starting level = 40%	25338	30827	29471
N$^{\underline{o}}$ of transmissions starting level = 50%	24518	31350	29426
N$^{\underline{o}}$ of transmissions starting level = 60%	23697	31873	28608
N$^{\underline{o}}$ of transmissions starting level = 70%	22877	32397	27790
N$^{\underline{o}}$ of transmissions starting level = 80%	22056	32920	26972
N$^{\underline{o}}$ of transmissions starting level = 90%	21235	33443	26106
N$^{\underline{o}}$ of transmissions starting level = 100%	20415	**33966**	25415

Table 5.15: N$^{\underline{o}}$ of transmissions for $T = 300 \cdot 10^{-5}$ s.

The total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 28448$$

And for the Non-aware protocol:

$$N_{tx} = 855$$

The analysis for variable latency is presented next (Table 5.16 and Fig. 5.10).

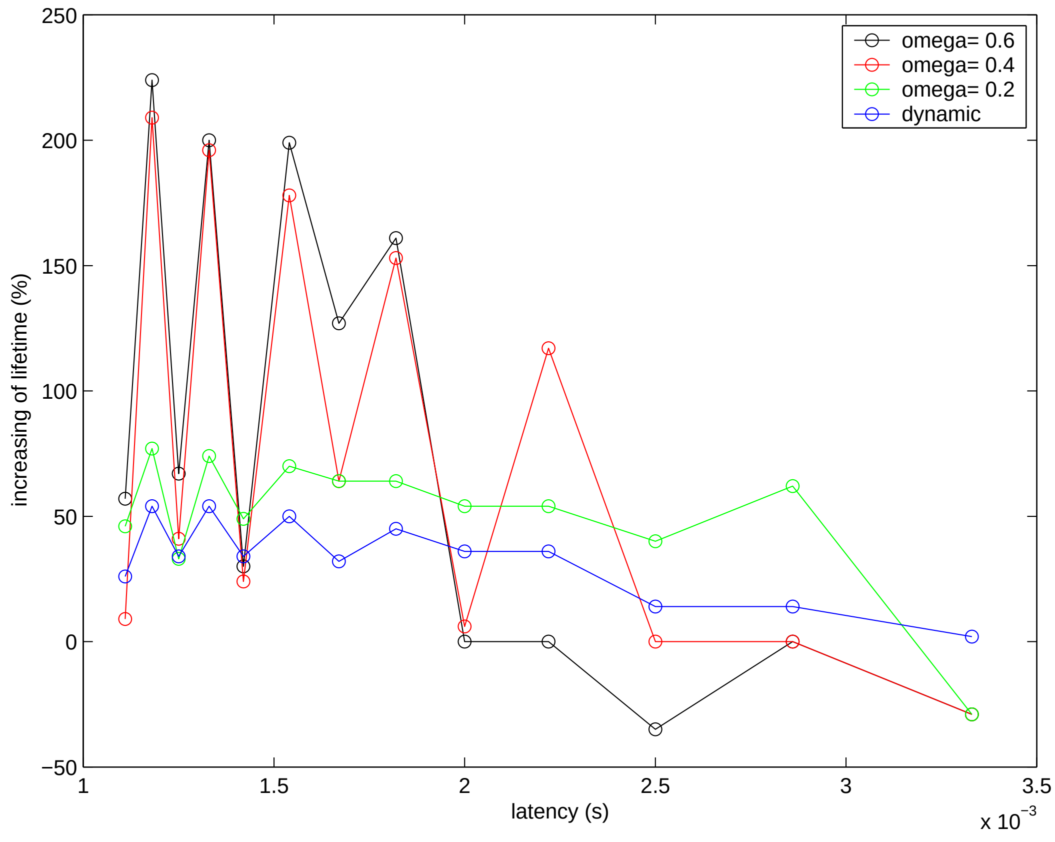

Figure 5.10: Performance for several latencies

As we have seen in other plots, the tendency of having less increase of lifetime as the latency increases is present here as well. If we compute the means we obtain the following performance:

- mean $(\Omega = 0.2) = 50.7782\%$
- mean $(\Omega = 0.4) = 74.4475\%$
- mean $(\Omega = 0.6) = 77.0181\%$
- mean $(dynamic) = 33.2\%$

If we use a combination of omegas we obtain the best choice:

- omega $= 0.6 \in T = \left[111 \cdot 10^{-5}, 200 \cdot 10^{-5}\right]$ s.
- omega $= 0.2 \in T = \left[200 \cdot 10^{-5}, 333 \cdot 10^{-5}\right]$ s.

Then we obtain:

- mean $(\Omega_{\text{MIXED}}) = 96.0413\%$ (almost 2 times higher)

$\Omega = 0.6$			$\Omega = 0.4$			$\Omega = 0.2$			*dynamic*		
b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)
9	*111*	57	*9*	*111*	9	*9*	*111*	46	*9*	*111*	26
8.5	*118*	224	*8.5*	*118*	209	*8.5*	*118*	77	*8.5*	*118*	54
8	*125*	67	*8*	*125*	41	*8*	*125*	33	*8*	*125*	34
7.5	*133*	200	*7.5*	*133*	196	*7.5*	*133*	74	*7.5*	*133*	54
7	*142*	30	*7*	*142*	24	*7*	*142*	49	*7*	*142*	34
6.5	*154*	199	*6.5*	*154*	178	*6.5*	*154*	70	*6.5*	*154*	50
6	*167*	127	*6*	*167*	64	*6*	*167*	64	*6*	*167*	32
5.5	*182*	161	*5.5*	*182*	153	*5.5*	*182*	64	*5.5*	*182*	45
5	*200*	0	*5*	*200*	6	*5*	*200*	54	*5*	*200*	36
4.5	*222*	0	*4.5*	*222*	117	*4.5*	*222*	54	*4.5*	*222*	36
4	*255*	-35	*4*	*255*	0	*4*	*255*	40	*4*	*255*	14
3.5	*286*	0	*3.5*	*286*	0	*3.5*	*286*	62	*3.5*	*286*	14
3	*333*	29	*3*	*333*	-29	*3*	*333*	-29	*3*	*333*	2

Table 5.16: Analysis for several latencies

5.4.2 Exponential Distribution 2

The power distribution for the one hundred nodes path is presented in Fig. 5.11.

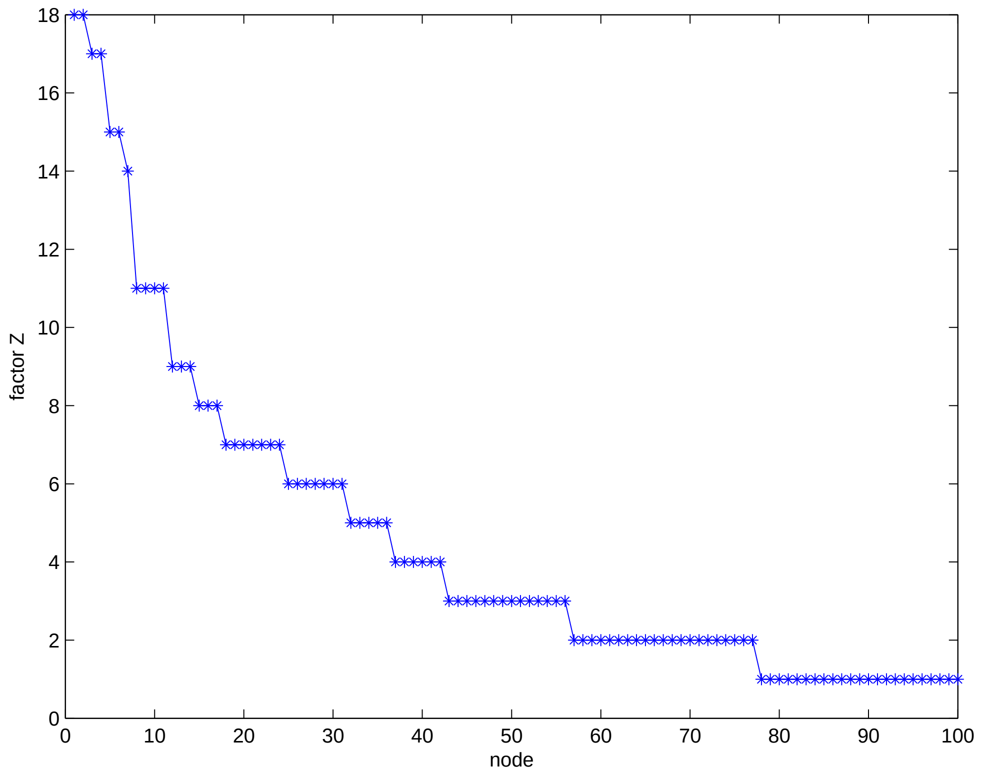

Figure 5.11: Consumption distribution along the path

The simulations gave us the results of the tables 5.17[5] and 5.18[6].
For the Table 5.17 the total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 2929$$

And for the Non-aware protocol:

$$N_{tx} = 482$$

For the Table 5.18 the total number of transmissions for the Power-aware protocol was:

$$N_{tx} = 15682$$

And for the Non-aware protocol:

$$N_{tx} = 482$$

[5]the parameter max_group is set to 9 nodes for $\Omega = 0.6$.
[6]the parameter max_group is set to 9 nodes for $\Omega = 0.4$.

Lifetime ($T = 150 \cdot 10^{-5}$s)	Balancing (static: $\Omega = 0.6$)	Balancing (static: $\Omega = 0.4$)	Balancing (dynamic)
N$^{\underline{o}}$ of transmissions starting level = 10%	3910	3455	3455
N$^{\underline{o}}$ of transmissions starting level = 20%	4640	3980	3704
N$^{\underline{o}}$ of transmissions starting level = 30%	5870	4506	3986
N$^{\underline{o}}$ of transmissions starting level = 40%	6478	4898	4188
N$^{\underline{o}}$ of transmissions starting level = 50%	6352	5135	4390
N$^{\underline{o}}$ of transmissions starting level = 60%	6446	5768	4372
N$^{\underline{o}}$ of transmissions starting level = 70%	6429	6429	4419
N$^{\underline{o}}$ of transmissions starting level = 80%	7755	5454	4219
N$^{\underline{o}}$ of transmissions starting level = 90%	7147	6399	4144
N$^{\underline{o}}$ of transmissions starting level = 100%	**7971**	6880	3981

Table 5.17: N$^{\underline{o}}$ of transmissions for $T = 150 \cdot 10^{-5}$s.

Lifetime ($T = 300 \cdot 10^{-5}$s)	Balancing (static: $\Omega = 0.4$)	Balancing (static: $\Omega = 0.2$)	Balancing (dynamic)
N$^{\underline{o}}$ of transmissions starting level = 10%	15682	16088	16264
N$^{\underline{o}}$ of transmissions starting level = 20%	15682	15611	16620
N$^{\underline{o}}$ of transmissions starting level = 30%	15682	15134	16977
N$^{\underline{o}}$ of transmissions starting level = 40%	15682	14657	17334
N$^{\underline{o}}$ of transmissions starting level = 50%	15682	14180	17555
N$^{\underline{o}}$ of transmissions starting level = 60%	15682	13702	**17583**
N$^{\underline{o}}$ of transmissions starting level = 70%	15682	13225	17554
N$^{\underline{o}}$ of transmissions starting level = 80%	15682	12748	17581
N$^{\underline{o}}$ of transmissions starting level = 90%	15682	12271	17552
N$^{\underline{o}}$ of transmissions starting level = 100%	15682	11794	17324

Table 5.18: N$^{\underline{o}}$ of transmissions for $T = 300 \cdot 10^{-5}$s.

From Table 5.17, we can state definitively that static parameters should choose a starting level of 100%. From Table 5.18, we see a lower performance for static parameters regarding the Power-aware protocol. As we will see in the following plot, our algorithm can work worse if latency is low. However, this problem can be solved by using a mix of omega constraints, depending on the current latency used.

The simulations for variable latency are shown in Table 5.19 and Fig. 5.12.

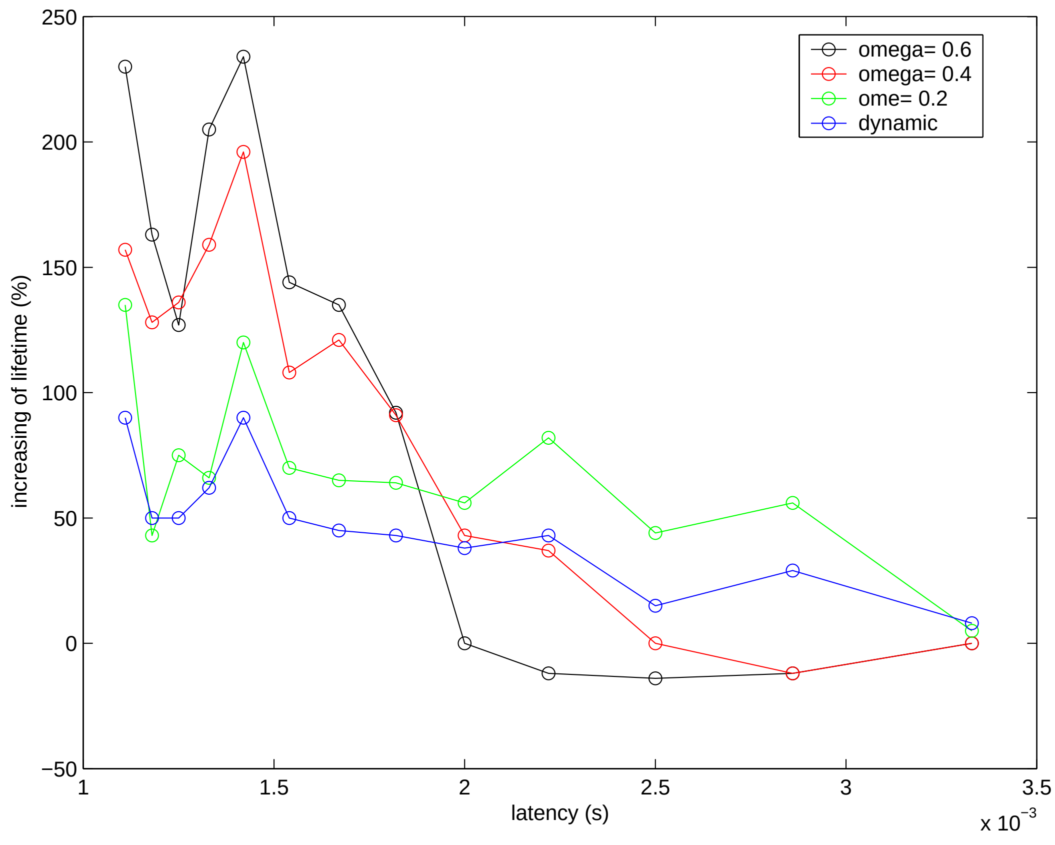

Figure 5.12: Performance for several latencies

If we compute the means we have:

- mean $(\Omega = 0.2) = 67.7877\%$
- mean $(\Omega = 0.4) = 89.6391\%$
- mean $(\Omega = 0.6) = 99.4455\%$
- mean $(dynamic) = 47.2692\%$

If we use a combination of omegas we obtain the best choice:

- omega $= 0.6 \in T = \left[111 \cdot 10^{-5}, 182 \cdot 10^{-5}\right]$ s.
- omega $= 0.2 \in T = \left[182 \cdot 10^{-5}, 333 \cdot 10^{-5}\right]$ s.

Then we obtain:

- mean $(\Omega_{\text{MIXED}}) = 121.1531\%$ (almost 2 times higher)

5.4.3 Conclusions for exponential distributions

On the one hand, comparing the results with the tree topology (ten nodes) of the previous section, we observe better results. Therefore we can state that the higher the number of the nodes that form a path, the longer the lifetime we can achieve.

On the other hand, comparing both exponential distributions performed in this section, we can say that the higher the difference between the nodes with higher consumption and the nodes with lower one, the higher the lifetime increase we can obtain.

$\Omega = 0.6$			$\Omega = 0.4$			$\Omega = 0.2$			*dynamic*		
b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)	b	$T \cdot 10^{-5}$	Δ Lifetime (%)
9	*111*	230	*9*	*111*	157	*9*	*111*	135	*9*	*111*	90
8.5	*118*	163	*8.5*	*118*	128	*8.5*	*118*	43	*8.5*	*118*	50
8	*125*	127	*8*	*125*	136	*8*	*125*	75	*8*	*125*	50
7.5	*133*	205	*7.5*	*133*	159	*7.5*	*133*	66	*7.5*	*133*	62
7	*142*	234	*7*	*142*	196	*7*	*142*	120	*7*	*142*	90
6.5	*154*	144	*6.5*	*154*	108	*6.5*	*154*	70	*6.5*	*154*	50
6	*167*	135	*6*	*167*	121	*6*	*167*	65	*6*	*167*	45
5.5	*182*	92	*5.5*	*182*	91	*5.5*	*182*	64	*5.5*	*182*	43
5	*200*	0	*5*	*200*	43	*5*	*200*	56	*5*	*200*	38
4.5	*222*	-12	*4.5*	*222*	37	*4.5*	*222*	82	*4.5*	*222*	43
4	*255*	-14	*4*	*255*	0	*4*	*255*	44	*4*	*255*	15
3.5	*286*	-12	*3.5*	*286*	-12	*3.5*	*286*	56	*3.5*	*286*	29
3	*333*	0	*3*	*333*	0	*3*	*333*	5	*3*	*333*	8

Table 5.19: Analysis for several latencies

Chapter 6

CONCLUSIONS

At this point, many simulations have been performed and many results have been achieved. We started with a single path configuration and we finished with multihop network topologies. The aim of this chapter is to gather all the conclusions and present together a summary of all the performances obtained.

6.1 Main Findings and Results

We carried out a performance comparison between our protocol and the Power-aware and Non-power-aware methods. On the one hand, the results regarding the Non-power-aware algorithm were not commented due to the really huge increases of lifetime obtained. Lifetime was improved up to 40 times with our approach compared to Non-power aware algorithm, depending on the latency constraint applied. On the other hand, many comparisons regarding the Power-aware protocol were presented. A summary of the improved performance is shown in the Fig. 6.1. The increases of lifetime ranged from 50% to 120% more transmissions.

On all topologies tested, we observed a tendency of having less increase of lifetime as the latency grew up (modulation parameter decreased). The reason can be explained with the Fig. 6.2. As it can be seen, when we are working with low modulation parameters (b= 2,3,4) the curves become flatter. To make matters worse, the latency separation between different modulations is bigger. On the contrarily, when we are working with high modulation parameters (7,8,9), the latency separations between different modulations are smaller. Also, the curves have a higher inclination. These characteristics make it more difficult to reach a specific saving in the low modulation area than in the high modulation one and therefore, the performance decreases as the latency rises up.

We have to mention that to our surprise, the results for dynamic power constraints were worse than static constraints. The reason is due to the fact that dynamic constraints are more conservative than static ones and therefore more unused energy remains in the nodes after a path is "lost" due to power-outage in one or more nodes.

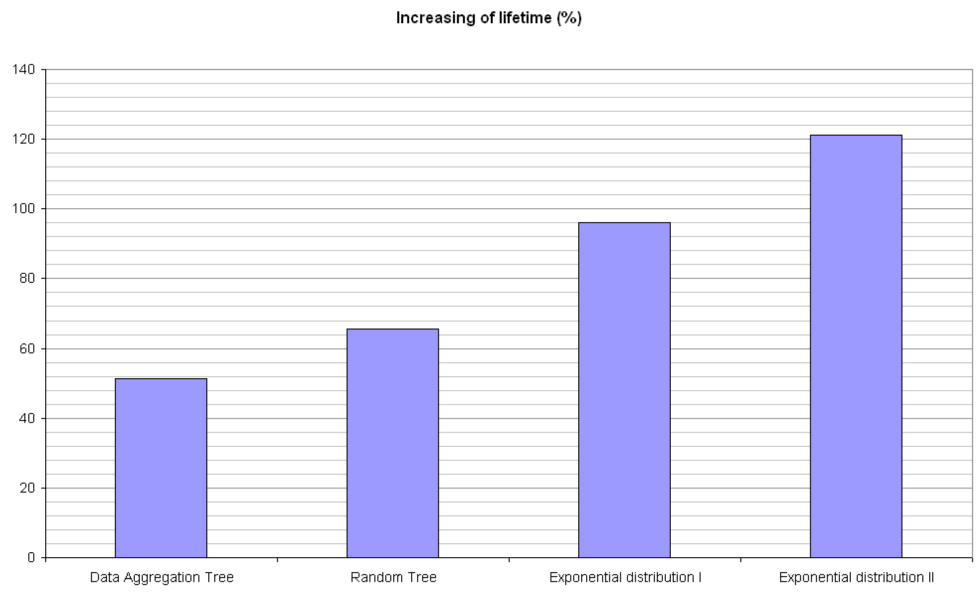

Figure 6.1: Performance summary

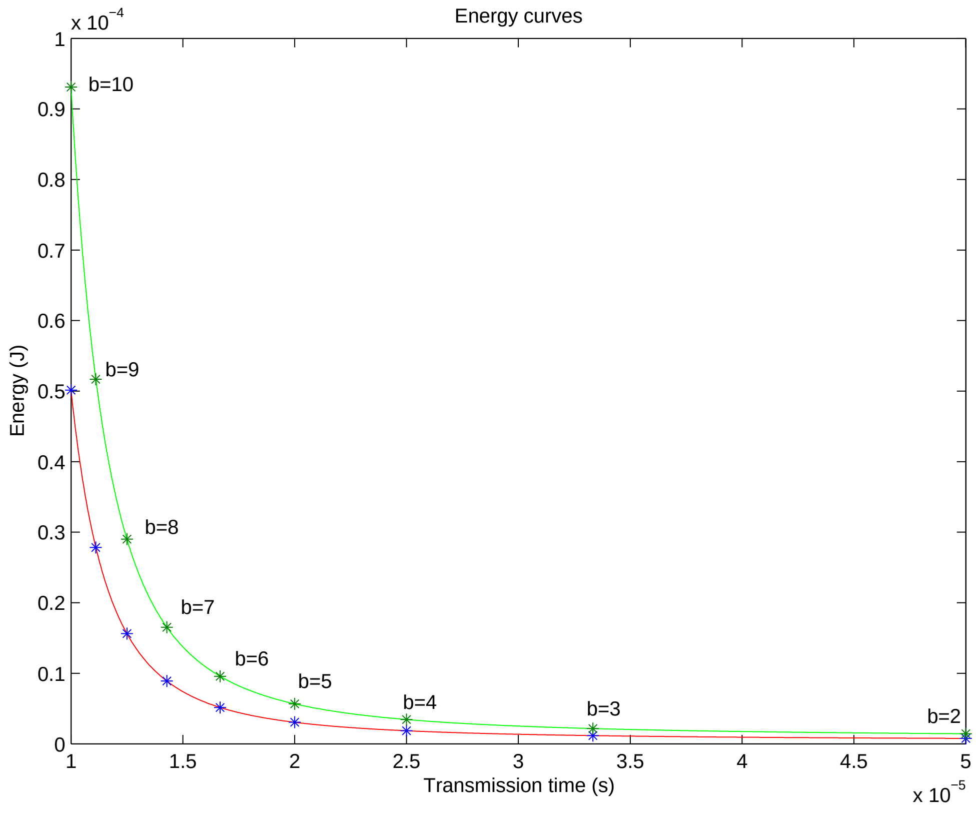

Figure 6.2: Energy curves

In addition, the fact that the improvements had a decreasing tendency as well when the latency increased made us to consider the possibility of combining different omegas. Therefore we found that it was efficient to use high omegas for low latencies (between 0.4 and 0.6) and low omegas for high latencies ($\Omega = 0.2$). Regarding the optimal value for gamma, we found that it should be $\gamma = 3$.

Furthermore, we realised that our algorithm should be executed since the beginning of the first transmission, i.e., when all the nodes have its battery level at 100%.

Moreover, we observed two important aspects:

- We got better results with longer paths than with shorter ones. The reason relies on the fact that we can use longer balancing groups and therefore impose higher omega constraints.
- The higher the consumption differences in a path of nodes, the higher the increase of lifetime of our algorithm.

We need to comment that the best combination of energy constraints has a strong dependence of the application that is being used. For instance, applications which work with high modulations parameters (low latency) do not require mixing different omegas. However if we have an application that is continuously changing the latency constraint for each data transmission, the mix of constraints must be performed.

Emphasize that all the simplifications and assumptions taken in our energy model were the same ones as the ones taken in the Power-aware algorithm. Also, the assumptions taken for our simulator were based on the same ones, with the only difference that we considered multiple paths crossing same node. This last consideration can be considered that we approach the problem in a biased way, giving more advantage to our protocol, but it is not true. In fact, what we do is to perform a more realistic simulation of the network, and therefore, the obtained increase of performance respect to the Power-aware protocol is valid.

All in all we conclude that the expected power savings and increased path lifetimes were found by the model simulations and analysis, and that the idea behind the algorithm ("altruism") of groups of nodes "helping" the "weaker" ones was favourable for all nodes.

6.2 Future Work

All the results given in this Master Thesis are based in our own simulator, which has been designed taking care about only a few aspects. Therefore, we should evaluate our algorithm in the physical layer of some professional simulator. An example could be the commercial distribution of QUALNET, which provides many protocols for different layers and many traffic patterns in order to simulate with precision our mechanism. Finally mention that a great challenge would be to implement our method in a real wireless sensor network such we obtained the real performance of it.

Bibliography

[1] Bhaskar Krishnamachari , *Networking Wireless Sensors* (University of Southern California: Cambridge University Press, 2005)

[2] Mostafa I. Abd-El-Barr, Mohamed A. Youssef and Maryam M. Al- Otaibi, *Wireless Sensor Networks - Part I: Topology and Design Issues*, 18th IEEE Annual Canadian Conference on Electrical and Computer Engineering CCECE 2005.

[3] D. Son, B. Krishnamachari, y J. Heidemann, "Experimental study of the effects of transmission power control and blacklisting in wireless sensor networks," *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on* (2004): 289-298.

[4] Jerry Zhao y Ramesh Govindan, "Understanding packet delivery performance in dense wireless sensor networks," en *Proceedings of the 1st international conference on Embedded networked sensor systems* (Los Angeles, California, USA: ACM, 2003), 1-13, http://portal.acm.org/citation.cfm?id=958491.958493.

[5] D. Ganesan et al., "Complex behavior at scale: An experimental study of low-power wireless sensor networks," *UCLA Computer Science Technical Report UCLA/CSD-TR* (2003): 02-0013.

[6] M. Srivastava, "Power-aware communication systems," *Power Aware Design Methodologies.*

[7] Y. Xu, J. Heidemann, y D. Estrin, "Geography-informed energy conservation for Ad Hoc routing," *Proceedings of the 7th annual international conference on Mobile computing and networking* (2001): 70-84.

[8] C. Schurgers, V. Tsiatsis, y M. B. Srivastava, "STEM: Topology management for energy efficient sensor networks," *Aerospace Conference Proceedings, 2002. IEEE* 3 (2002).

[9] Jae-Hwan Chang y L. Tassiulas, *"Energy conserving routing in wireless ad-hoc networks," en INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings.* IEEE, 2000, 22-31 vol.1.

[10] W. R. Heinzelman et al., "Energy-efficient communication protocol for wireless microsensor networks," System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on (2000): 10.

[11] Wei Ye, J. Heidemann, y D. Estrin, “An energy-efficient MAC protocol for wireless sensor networks,” en INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 2002, 1567-1576 vol.3.

[12] Barbara Hohlt, Lance Doherty, y Eric Brewer, “Flexible power scheduling for sensor networks,” en Proceedings of the third international symposium on Information processing in sensor networks (Berkeley, California, USA: ACM, 2004), 205-214

[13] Curt Schurgers, Olivier Aberthorne, y Mani Srivastava, “Modulation scaling for Energy Aware Communication Systems,” en Proceedings of the 2001 international symposium on Low power electronics and design (Huntington Beach, California, United States: ACM, 2001), 96-99

[14] C. Schurgers, V. Raghunathan, y M.B. Srivastava, “Modulation scaling for real-time energy aware packet scheduling,” en Global Telecommunications Conference, 2001. GLOBECOM ’01. IEEE, 2001, 3653-3657 vol.6.

[15] C. Schurgers y M.B. Srivastava, “Energy efficient wireless scheduling: adaptive loading in time,” en Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE, 2002, 706-711 vol.2.

[16] Zongkai Yang, Yong Yuan, y Jianhua He, “Energy aware data gathering based on adaptive modulation scaling in wireless sensor networks,” en Vehicular Technology Conference, 2004. VTC2004-Fall. 2004 IEEE 60th, 2004, 2794-2798 Vol. 4.

[17] Y. Yu y V. K. Prasanna, “Energy-Efficient Multi-Hop Packet Transmission using Modulation Scaling in Wireless Sensor Networks,” energy 1: 2.

[18] Proakis, J., “Digital Comunications,” McGraw-Hill Series in Electrical and Computer Engineering, 3rd Edition, 1995.

[19] Andrew Wang et al., “Energy efficient Modulation and MAC for Asymmetric RF Microsensor Systems,” en Proceedings of the 2001 international symposium on Low power electronics and design (Huntington Beach, California, United States: ACM, 2001), 106-111.

[20] Xiaobing Wu, “Avoiding Energy Holes in Wireless Sensor Networks with Nonuniform Node Distribution,” text, May 2008.

[21] Fan Yong et al., “Energy Consumption Distribution-Aware Node Placement in Wireless Sensor Networks (WSNs),” en Wireless Communications, Networking and Mobile Computing, 2006. WiCOM 2006.International Conference on, 2006, 1-4.

[22] Y. Yu, B. Krishnamachari, y V. K. Prasanna, “Energy-latency tradeoffs for data gathering in wireless sensor networks,” IEEE INFOCOM 1 (2004): 244-255.

Appendix A

Matlab Code for the Algorithm

A.1 Main.m

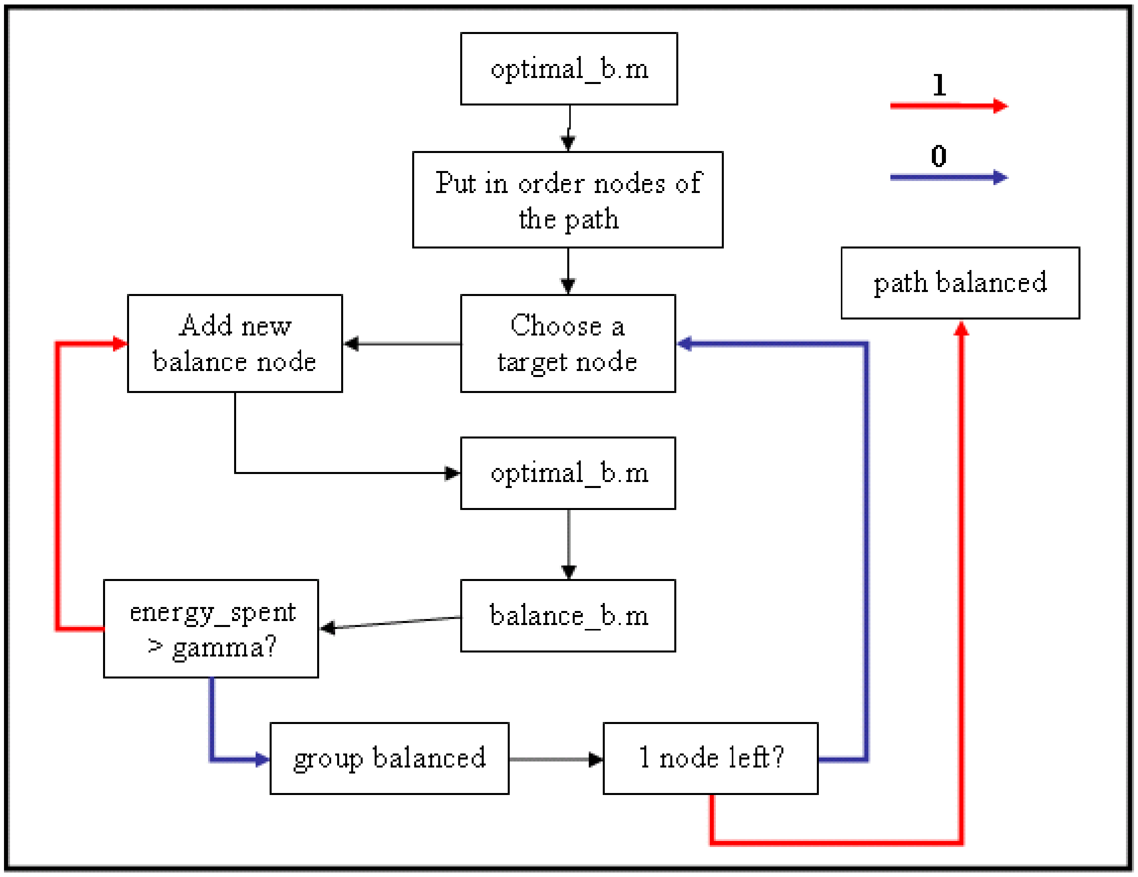

Figure A.1: Flow diagram of main.m

```
function [node,groups_balanced,not_balanced]=main(C,D,Lat,R,s,node_energy)
%------------------INPUTS----------------------
%C is a vector(row) that symbolizes the tx power
%D is a vector(row) that symbolizes the electronic power
%R is the transmission rate
%s is the packet size
%Lat total path latency
%node_energy is the energy of each node
%------------------OUTPUTS---------------------
%groups_balanced is a vector that contains the positions of
%a balanced node group in the vector node which has been
%put in order from lower batt til higher batt not_balanced
%are the nodes not balanced due to several reasons
%----------------------------------------------
max_energy=10; %in Jules
max_group=5;
%structure of node
node=[];
node.index=0;
node.C=0;           %tx power parameter
node.D=0;           %electronic parameter
node.b=0;           %mod parameter balanced
node.b0=0;          %theoretical optimal mod param
node.bop=0;         %theoretical optimal ceiled mod param
node.T=0;           %latency of node i of the path
node.energy=0;      %remaining battery level
node.Eused=0;       %energy used
node.ratio=0;
node.omega=0;    %of energy that we want to save if it's
                 %target node
node.gamma=0;    %max over-power-spending factor if it's
                 %balancing node
%compute the optimal Energies and tx times for each node
%if we consider the whole path
[E0,T0]=optimal_energy(Lat,length(C),C,D,R,s,0.001);
b0=s./(T0.*R);
[bop,nn,nnn,mm,mmm,kk]=optimal_b(Lat,C,D,R,s);
%path creation
for i=1:length(C)
    node(i).index=i;
    node(i).energy=node_energy(i);
    node(i).b=10;
    node(i).b0=b0(i);
    node(i).bop=bop(i);
    node(i).C=C(i);
    node(i).D=D(i);
    node(i).T=T0(i);
    %energy constraints taking into account battery level

    [node(i).omega,node(i).gamma]=energyconstraints(node(i) .energy,max_energy);
end
%PUT IN ORDER NODES IN THE PATH
%from lower energies to higher ones
for i=2:(length(node))
    for j=1:(length(node)-1)
        if node(j).energy > node(j+1).energy
            aux = node(j);
            node(j)= node(j+1);
            node(j+1)= aux;
        end
    end
end
%PERFORM BALANCING IN ONE PATH
avaiable_nodes=length(node);
groups_balanced=0;
not_balanced=[];
adjust=0;
j=1;
%++++++++++++++++++++++++beguin of for+++++++++++++++++++++++++++++++++++++
for k=1:length(node)
disp('####################################################################')
disp('####################################################################')

    disp('starting a new balancing group of nodes...')

    flag=0;
    selected_nodes=[];
    selected_C=node(k).C;
    selected_D=node(k).D;
    selected_nodes=[selected_nodes k];
    avaiable_nodes=avaiable_nodes-1;

    %when there's 1 node alone:
    if avaiable_nodes == 0
```

```
        aux=0;
        for i=1:length(node)
            aux=aux+node(i).T;
        end
        aux=aux-node(k).T;
        node(k).T=Lat-aux; %max tx time allowed for last node unpaired

        node(k).b=ceil(s/((node(k).T)*R)); %round toward inf to satisfy lat
        node(k).T=s/((node(k).b)*R);

        aux=(((node(k).C)*((2.^(node(k).b))-1)+(node(k).D)).*s./(node(k).b));
        node(k).energy=(node(k).energy)-aux;
        node(k).Eused=aux;
        node(k).ratio=aux/E0;
        %updating omega and gamma
        [node(k).omega,node(k).gamma]=energyconstraints(node(k).energy,max_energy);

        not_balanced=[not_balanced k];
        break %breaking for

    elseif avaiable_nodes < 0
        break %breaking for
    end
    %end of alone node

  %************************beguin of%while***************************
%*******************************************************************
%this is the block that adds a new node to the group of nodes that are
%trying to save power in the target node.
    while flag==0
    disp('--------------------------------------------------------------')


    %Blocking node problem: only allowed to compute together
    %less or equal than max_grop nodes (target+ balance nodes)
        if length(selected_nodes) == max_group
            for i=1:length(selected_nodes)

                node(selected_nodes(i)).b=b(i);
                node(selected_nodes(i)).T=Topt(i);

                node(selected_nodes(i)).energy=node(selected_nodes(i)).energy-Eopt(i);
                node(selected_nodes(i)).Eused=Eopt(i);
                node(selected_nodes(i)).ratio=Eopt(i)/E0;

                %updating omega and gamma
                [node(selected_nodes(i)).omega,node(selected_nodes(i)).gamma]=↙
energyconstraints(node(selected_nodes(i)).energy,max_energy);

                fprintf('node:');
                fprintf('  %d  ',selected_nodes(i));
                fprintf('not balanced due to max number of nodes↙
in balance group was overcame\n');
            end

            j=j-1; %due to we come from GAMMA ERROR & ERROR1 and
                   %they add 1 to j in order
                   %to add a new node to the group. So we have
                   %to decrease in 1 unit j
            not_balanced=[not_balanced selected_nodes];
            break %breaking while
        end
       %here only goes in when more than 1 balancing node is
       %required but we do not have more than 1 balancing node
       %avaiable. Ex: we have 2 nodes left, 1 is the target,
       %the other the balancing node. So if we cannot perform
       % we are going to need another one but
       %balancing there are not more avaiable. So both nodes
       %are updated here.
        if avaiable_nodes ==0
            for i=1:length(selected_nodes)

                node(selected_nodes(i)).b=b(i);
                node(selected_nodes(i)).T=Topt(i);
                node(selected_nodes(i)).energy=↙
node(selected_nodes(i)).energy-Eopt(i);
                node(selected_nodes(i)).Eused=Eopt(i);
                node(selected_nodes(i)).ratio=Eopt(i)/E0;

                %updating omega and gamma
                [node(selected_nodes(i)).omega,node(selected↙
_nodes(i)).gamma]=energyconstraints(node(selected_nodes(i)).↙
energy,max_energy);
```

```
            fprintf('node:');
            fprintf('  %d  ',selected_nodes(i));
            fprintf('not balanced \n');
        end

        not_balanced=[not_balanced selected_nodes];
        break %breaking while
    end
  %selection of one node more
    selected_C=[selected_C node(length(node)+2-k-j+adjust).C];
    selected_D=[selected_D node(length(node)+2-k-j+adjust).D];
    selected_nodes=[selected_nodes (length(node)+2-k-j+adjust)];
    avaiable_nodes=avaiable_nodes-1;



    %##########################################################
    %###################PERFORM THE BALANCE###################
    tot_Lat=0;
    for i=1:length(selected_nodes)
        tot_Lat=tot_Lat+node(selected_nodes(i)).T;
    end
    %++++++++++++++++IMPORTANT++++++++++++++++++++++++++++++++++++++++

    %Now the order of the power parameters parameters is:
    %C=[C(target node) C(group node1) C(group node2) ...]
    %Where the associated battery levels are
    %Lgroupnode1 > Lgroupnode2 > ... > Ltargetnode

    %in order to execute easier the following next two functions we have
    %to organize the parameters C and D. So in C and D, the lower the index
    %the higher the battery level.
    %For instance: C(1) must be a power parameter of a node with 90% battery
    %              C(7) must be a power parameter of a node with 15% battery
    %              C(length) must be pow param of target node (10% of battery)
    %Always C(length(C)) will be the power parameter of the TARGET NODE

    aux=selected_C(1);
    selected_C(1:(length(selected_C)-1))=selected_C(2:length(selected_C));
    selected_C(length(selected_C))=aux;

    aux=selected_D(1);
```

```
        selected_D(1:(length(selected_D)-1))=selected_D(2:length(selected_D));
        selected_D(length(selected_D))=aux;

        %+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        [b,Eopt,notusedb0,notusedE0,notusedE_S,Topt]=↙
optimal_b(tot_Lat,selected_C,selected_D,R,s);
        E_S=Eopt./E0;
        %selected_nodes
        %b

        [bb,energy_spent,Ebalanced,Tn]=balance_b(b,E0,E_S,↙
node(selected_nodes(1)).omega,selected_C,selected_D,tot_Lat,R,s);
        %energy_spent
        %bb

        %balance_b can give 2 errors:
        %ERROR1:when the overspending is so hight: energy_spent = Inf
        %ERROR2:when omega can't be achieved: energy_spent = -1

        %++++++++++++++++IMPORTANT+++++++++++++++++++++++++++++++++++++++++
        %Now we have to reorder again the outputs of the balance_b.m
        %From [balancenode1 balancenode2 ...targetnode]
        %to [targetnode balancenode1 balancenode2 ....]
        aux=b(length(b));
        b(2:length(b))=b(1:(length(b)-1));
        b(1)=aux;

        aux=Eopt(length(Eopt));
        Eopt(2:length(Eopt))=Eopt(1:(length(Eopt)-1));
        Eopt(1)=aux;

        aux=Topt(length(Topt));
        Topt(2:length(Topt))=Topt(1:(length(Topt)-1));
        Topt(1)=aux;

        aux=bb(length(bb));
        bb(2:length(bb))=bb(1:(length(bb)-1));
        bb(1)=aux;

        aux=energy_spent(length(energy_spent));
        energy_spent(2:length(energy_spent))=energy_spent(1:↙
(length(energy_spent)-1));
```

```
        energy_spent(1)=aux;

        aux=Ebalanced(length(Ebalanced));
        Ebalanced(2:length(Ebalanced))=Ebalanced(1:(length(Ebalanced)-1));
        Ebalanced(1)=aux;

        aux=Tn(length(Tn));
        Tn(2:length(Tn))=Tn(1:(length(Tn)-1));
        Tn(1)=aux;

        aux=selected_C(length(selected_C));
        selected_C(2:length(selected_C))=selected_C(1:(length(selected_C)-1));
        selected_C(1)=aux;

        aux=selected_D(length(selected_D));
        selected_D(2:length(selected_D))=selected_D(1:(length(selected_D)-1));
        selected_D(1)=aux;

        %#################################################################
        %####################END OF BALANCE################################
    %####################HANDLE ERRORS###############################

        %ERROR2
        %We have to free the node that is associated to the 1st one and
        %also we have to update the 1st node to the optimal case
        if energy_spent==-1
            disp('cannot reach omega savings')
            %for i=1:length(selected_nodes)
                i=1;
                node(selected_nodes(i)).b=ceil(node(selected_nodes(i)).b0);

                Topt2=s./((node(selected_nodes(i)).b).*R);
                node(selected_nodes(i)).T=Topt2;

                node(selected_nodes(i)).Eused = energy_model2(node(sele↙
cted_nodes(i)).C,node(selected_nodes(i)).D,R,s,node(selected_nodes(i)).b);
                node(selected_nodes(i)).energy=node(selected_nodes(i)).↙
energy-node(selected_nodes(i)).Eused;

                node(selected_nodes(i)).ratio=node(selected_nodes(i)).Eused/E0;

                adjust=adjust+1;
                avaiable_nodes=avaiable_nodes+1;

                [node(selected_nodes(i)).omega,node(selected_nodes(i))↙
.gamma]=energyconstraints(node(selected_nodes(i)).energy,max_energy);

                not_balanced=[not_balanced selected_nodes(i)];

                fprintf('node:');
                fprintf('  %d  ',selected_nodes(i));
                fprintf('not balanced because unable to achieve↙
omega= %d \n',node(selected_nodes(1)).omega);
            %end
            break %breaking while
        end
          %GAMMA ERROR & ERROR1
        %if we overcome the spending limit in some node we increase the number
        %of nodes to help to satisfy the overspending constraint (<gamma)
        flag=1;
        disp('checking gamma...')
        for i=1:length(selected_nodes)
            if energy_spent(i)>node(selected_nodes(i)).gamma
                if i==1  %error1
                    fprintf('energy savings cannot be reached with %d↙
balancing nodes \n',(length(selected_nodes)-1));
                    disp('adding one node more to the balancing group...')
                    j=j+1;
                    flag=0;
                    break %breakin for
                else
                    fprintf('over spending constraint violated ↙
in %d nd/rd/th node\n',selected_nodes(i));
                    disp('adding one node more to the balancing group...')
                    j=j+1;
                    flag=0;
                    break %breakin for
                end
            end
        end
       %################END OF HANDLE ERRORS###########################

        %if flag==1 means that every constraint are satisfied and the group
        %of nodes are balanced so, it's the moment for updating the new values
        if flag==1
            disp('all constraints satisfied: updating values...')
            groups_balanced=[groups_balanced selected_nodes 0];
            for i=1:length(selected_nodes)

                node(selected_nodes(i)).b=bb(i);
                node(selected_nodes(i)).T=Tn(i);
                node(selected_nodes(i)).energy=node(selected_nodes(i)).energy↙
-Ebalanced(i);
                node(selected_nodes(i)).Eused=Ebalanced(i);
                node(selected_nodes(i)).ratio=Ebalanced(i)/E0;

                [node(selected_nodes(i)).omega,node(selected_nodes(i)).gamma]=↙
energyconstraints(node(selected_nodes(i)).energy,max_energy);
                %disp('selected nodes:')
                %fprintf('  %d  ',selected_nodes(i));
                %fprintf('\n');
            end
        end
    end
%******************************************************************************
%********************************end of while**********************************


end
%++++++++++++++++++++++++end of for+++++++++++++++++++++++++++++++++++++
%STATISTICS
time=0;
for i=1:length(node)
    ratio(i)=node(i).ratio*100;
    b(i)=node(i).b;
    b0(i)=node(i).b0;
    bop(i)=node(i).bop;
    time=time+node(i).T;
end
time=time
x=linspace(1,length(node),length(node));
subplot(1,2,1);
plot(x,b,'bo',x,b,'b',x,b0,'ro',x,b0,'r');
xlabel('node number(higher numbers are nodes with more remaining energy)')
ylabel('modulation parameter')
title('plot of balanced modulation parameters and theoretical optimal ones')
```

```
subplot(1,2,2);
plot(x,ratio,'bo',x,ratio,'b');
xlabel('node number(higher numbers are nodes with more remaining energy)')
ylabel('% energy spent')
title('plot of the % of energy spent regarding to the optimal energy')

%PUT IN ORDER THE NODES AGAIN FROM
%LOWER INDEX TO HIGHER ONE
for i=2:(length(node))
    for j=1:(length(node)-1)
        if node(j).index > node(j+1).index
            aux = node(j);
            node(j)= node(j+1);
            node(j+1)= aux;
        end
    end
end

%end of main
```

A.2 Energyconstraints.m

```
function [ome,gam]=energyconstraints(energy,max_energy)
%This function tries to choose the proper energy parameters
%of each node.
%The decisions are based on the battery level of each node.
batlevel=(energy)*100/max_energy;
%###################
%dynamic I
%###################
%if batlevel < 15
%    ome=0.4;
%    gam=1;
%elseif batlevel > 50
%    ome=-2;
%    gam=3;
%else
%    ome=0.15;
%    gam=2.7;
%end
%for latencies around 3e-5 for each node:
%if batlevel < 15
%    ome=0.2;
%    gam=1;
%elseif batlevel > 50
%    ome=-2;
%    gam=3;
%else
%    ome=0.1;
%    gam=2.7;
%end

%###################
%static constraints
%###################
ome=0.4;
gam=3;
%for latencies around 3e-5 for each node:
%ome=0.2;
%gam=3;
%end of energyconstraints
```

A.3 Optimal_energy.m

```
function [Eopt,Tn]=optimal_energy(T,n,C,D,R,s,PR)
%----------------------------------------------------------------
%This algorithm uses binary search for achieve a solution
%to compute the optimal energy for each node when all the
%nodes have the same energy levels.
%Based on paper: "Energy-Efficient Multi-Hop Packet Transmission
%using Modulation Scaling in Wireless Sensor Networks"
%---------inputs---------------------------------------
%T is the delay constraint in our system
%n is the number of transmitting nodes in the path
%C is a vector(row) that symbolizes the tx power
%D is a vector(row) that symbolizes the electronic power
%s is the packet length in bits
%PR is the precision when searching for the latency constraint
%from T*(1-E) to T*(E+1) --> values of E=from 0.00001 to 0.0001
%--------------------------------------------------------
%------------outputs------------------------------------
%Eopt is the optimal energy that satisfies the constraints
%Tn is a vector that contains the latency of each node
%--------------------------------------------------------
%conditions for optimality: E(i)=E(i+1) for i=0,1,..,n-1;
%                           sum(Tn)=T;
%-------------------------------
E=zeros(1,n);  %energy spent of each node
Tn=zeros(1,n); %latency for each node
for i = 1:n,
    Tn(i)=T./n;
end
%compute the Emin and Emax for t=T/n
for i = 1:n,
    E(i)=energy_model(C(i),D(i),R,s,Tn(i));
end
Emax=max(E);
Emin=min(E);
%checking if we have already the solution of the problem
j=0;
for i = 1:(n-1),
    if E(i)~=E(i+1)
        j=1;
    end
```

```
end
%if j=0 we have finished

%main body
%1st-> solve for the new Eopt the latency constraints
%2nd check if is optimal
%3rd if not optimal perform binary search
if j==1
    b=zeros(1,n);
    Tn=zeros(1,n);
    Ts=sum(Tn);
    Eopt=0;
else
    Eopt=E(1);
    %Tn=Tn;
end
while j==1,

         Eopt=(Emax+Emin)./2;

         for i = 1:n
             E(i)=Eopt;
             b(i)=fixedpoint('energy_fixedpoint',8,1e-6,1000,C(i),D(i),Eopt,s);
         end


         Tn=s./(b.*R);
         Ts=sum(Tn);
         if (Ts > (T*(PR+1)))   %not optimal
             Emax=Emax;
             Emin=Eopt;
         elseif (Ts < (T*(1-PR)))   %not optimal
             Emax=Eopt;
             Emin=Emin;
         else   %is optimal
             j=0;
         end

end
%end of optimal_energy
```

A.4 Optimal_b.m

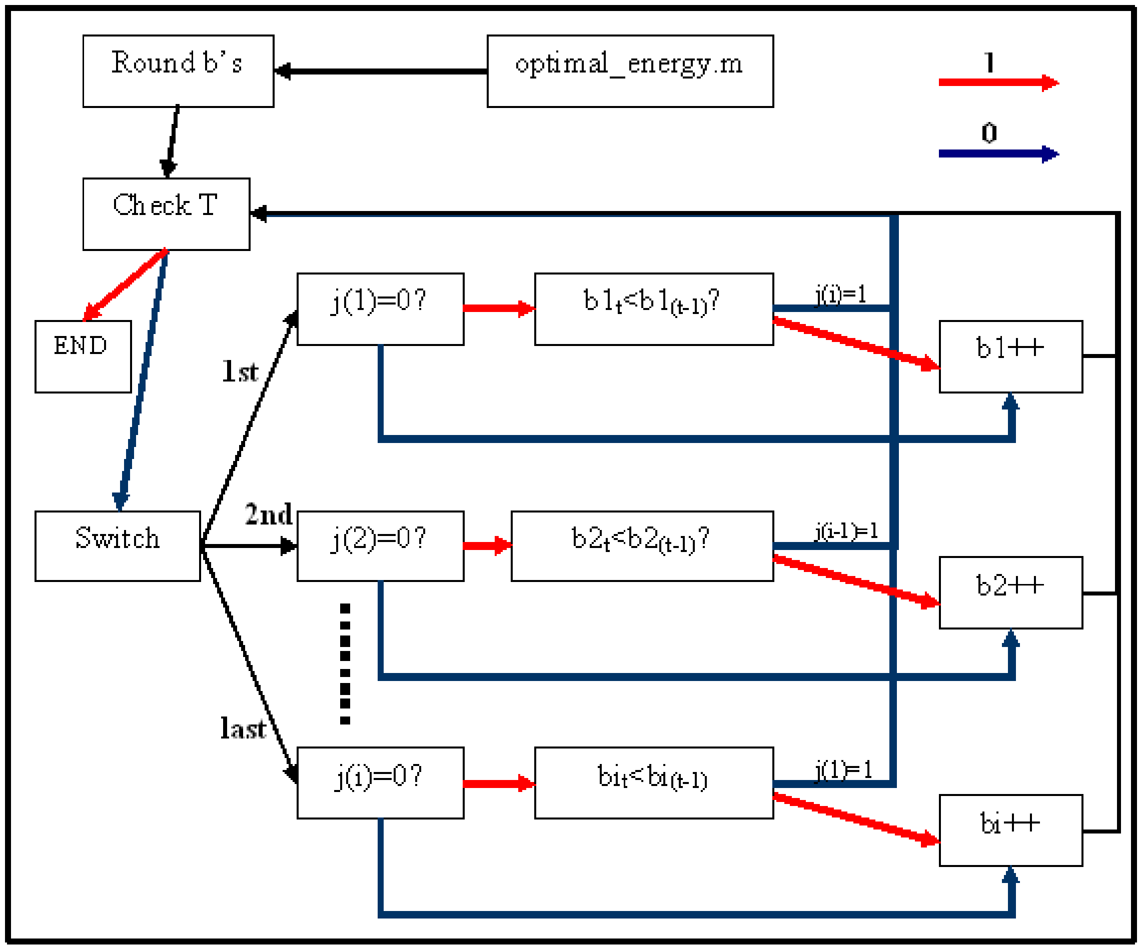

Figure A.2: Flow diagram of optimal_b.m

```
function [b,Eopt,b0,E0,E_S,Tn]=optimal_b(T,C,D,R,s)
%IMPORTANT: In C and D, the lower the index the higher the battery level
%For instance: C(1) is a power parameter of a node with 90% battery
%              C(7) is a power parameter of a node with 15% battery
%******************INPUTS*****************************************
%C is a vector(row) that symbolizes the tx power
%D is a vector(row) that symbolizes the electronic power
%R is the transmission rate
%s is the packet size
%T total path latency
%*********************OUTPUTS*************************************
%b (vector) is the optimal modulation parameter ceiled
%Eopt (vector) are the optimal energies for nodes with ceiling
%E_S is the energy ratio spending for nodes with rounded b's respect the theoric optimal
%b0 (vector) is the optimal parameter before ceiling
%E0 (vector)are the optimal energies before ceiling
%Tn are the latencies asociated to b's
%*****************************************************************
%COMPUTE OPTIMAL CONSUMPTION
%disp('####################################################################')
%disp('####################################################################')
%Theoretical
disp('computing theoretical optimal energy...')
[E0,Tn]=optimal_energy(T,length(C),C,D,R,s,0.001);
%disp('computing ceiling...')
%Realistic(ceiling function)
b0=s./(Tn.*R);
b=round(b0);
Tn=s./(b.*R);
Ts=sum(Tn);
%the aim is to round the modulation parameters
%in order to achieve lower increments of b in nodes
%with less remaining energy.
%j=length(C); wrong
j=1;
nodeflag=zeros(1,length(C));
while Ts > (T*(0.001+1))
    if nodeflag(j)==0
        if b(j)<b0(j)
%            fprintf('increasing modulation parameter of node %d \n',j)
            b(j)=b(j)+1;
        else
```

```
            nodeflag(j)=-1;
        end
    else
        b(j)=b(j)+1;
    end


    Tn=s./(b.*R);
    Ts=sum(Tn);

    %j=j-1; wrong
    %if j==0
    %    j=length(C);
    %end

    j=j+1;
    if j==(length(C)+1)
        j=1;
    end

end
for i = 1:length(C)
    Eopt(i)=((C(i).*((2.^b(i))-1)+(D(i))).*s./b(i));
end
%disp('ceiling finished')
E_S=Eopt./E0;
%for i = 1:length(C)
%    fprintf('percentage of energy spent in node %d: respect the↙
theoric optimal is %f  \n',i,(E_S(i).*100))
%end
%disp('####################################################################')
%disp('####################################################################')
%end of optimal_b
```

A.5 Balance_b.m

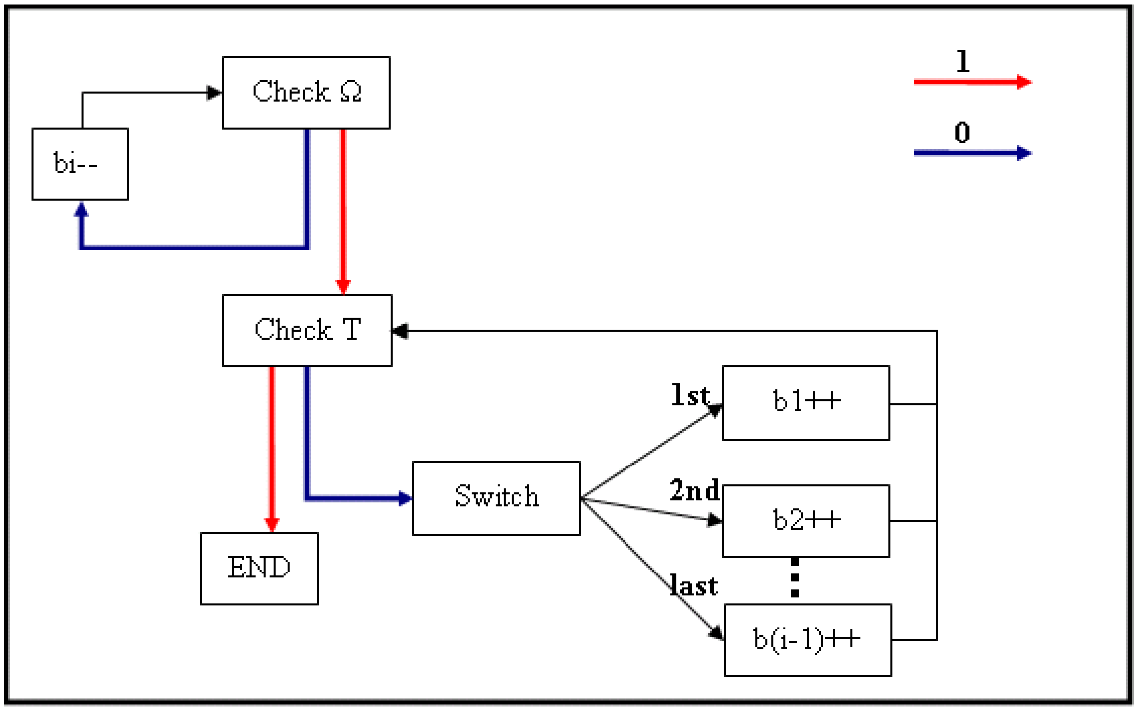

Figure A.3: Flow diagram of balance_b.m

```
function [bb,energy_spent,Eopt,Tn]=balance_b(b,E0,E_S,omega,C,D,T,R,s)
%IMPORTANT: In C and D, the lower the index the higher the battery level
%For instance: C(1) is a power parameter of a node with 90% battery
%              C(7) is a power parameter of a node with 15% battery
%********************INPUTS******************************************
%b is the optimal ceiled modulation parameters
%E0 is the theoretical optimal spending energy in each node
%E_S is the energy ratio spent respect the theoric optimal
%omega is the amount of energy that we want to save in node 1
% omega € [0,1]
%C is a vector(row) that symbolizes the tx power
%D is a vector(row) that symbolizes the electronic power
%R is the transmission rate
%s is the packet size
%T total path latency
%***********************OUTPUTS**************************************
%bb (vector) is the optimal modulation parameter ceiled
%energy_spent is the energy ratio spent respect the theoretical optimal
%Eopt (vector) are the optimal energies for balanced nodes with ceiling
%Tn are the tx times for balanced nodes with ceiling
%********************************************************************
%disp('####################################################################')
%disp('####################################################################')
disp('computing balance...')
%bmax is the limit considered for not falling in an infinite bucle
%when performing T check.
bmax=10;
err=0;
energy_spent=E_S;
%CHECKING OMEGA(energy saved in node 1)
disp('checking omega...')
while (1-energy_spent(length(C))) < (omega*(0.001+1))
    b(length(C))=b(length(C))-1;
%    disp('decreasin modulation parameter of target node')

    if b(length(C))==1
        disp('cannot reach energy savings in target node ...')
        err=-1;
        break
    end

    Eopt(length(C))=((C(length(C)).*((2.^b(length(C)))-1)+↙
(D(length(C)))).*s./b(length(C)));
    energy_spent(length(C))=Eopt(length(C))./E0;
end
if err==-1
    bb=-1;
    energy_spent=-1;
    Eopt=-1;
    Tn=-1;
    return
end
%CHECKING T
disp('checking T...')
Tn=s./(b.*R);
Ts=sum(Tn);
%j=length(C); this is wrong
j=1;
while Ts > (T*(0.001+1))
    b(j)=b(j)+1;
%    fprintf('increasing modulation parameter of node %d \n',j)
    Tn(j)=s./(b(j).*R);
    Ts=sum(Tn);

    if min(b(1:(length(b)-1)))>=bmax
        if Ts > (T*(0.001+1))
%            disp('scaping from infinite bucle...')
            %fprintf('energy savings in node 1 cannot be reached↙
with %d balancing node/s \n',(length(C)-1))
            err=-1;
            break
        end
    end


    %j=j-1; wrong
    j=j+1;
    if j==length(b)
        j=1;
    end

    %if j==1    wrong
    %    j=length(C);
    %end
```

```
end
bb=b;
if err==-1
    bb=b.*Inf;
    energy_spent=E_S.*Inf;
    Eopt=bb;
    Tn=bb;
    return
end
%ENERGY SPENDING RESULTS
for i = 1:length(C)
    Eopt(i)=((C(i).*((2.^b(i))-1)+(D(i))).*s./b(i));
end
energy_spent=Eopt./E0;
%disp('balance of b‘s finished')
%disp('####################################################################')
%disp('####################################################################')
%End of balance_b
```

A.6 Energy_model.m

```
function E=energy_model(C,D,R,s,t)
b=s./(t.*R);
E=(C.*((2.^b)-1)+(D)).*s./b;
```

Appendix B

Matlab Code for Simulator

B.1 Theoretical Path Simulator

```
function [node,n_tx,n_tx2,n_tx3,node_energy,node_energy2,node_energy3]=↙
simula(C,D,T,R,s,node_energy)
%INPUTS-----------------------------
%C is a vector(row) that symbolizes the tx power
%D is a vector(row) that symbolizes the electronic power
%R is the transmission rate
%s is the packet size
%T total path latency
%node_energy is the energy of each node
%OUTPUTS
%n_tx is the nº of transmission for our protocol
%n_tx2 the transmissions for the Non-Power-aware
%n_tx3 the transmissions for the Power-aware
%-----------------------------------
%rate determines in our algorithm the up-date frequency
%of the modulation parameters of the whole path
b=0;
rate=100;
n_tx=0;
n_tx2=0;
n_tx3=0;
node_energy2=node_energy;
node_energy3=node_energy;
%#######################NON POWER AWARE PROTOCOL###################
a=1;
Emax=energy_model2(C,D,R,s,10);
while a >= 0
        n_tx2=n_tx2+1;

        for i=1:length(C)
            node_energy2(i)=node_energy2(i)-Emax(i);
            if node_energy2(i) <= 0
                fprintf('NON-AWARE protocol:number of total tx was %d \n',n_tx2);
                a=-1;
            end
        end
end
%#######################POWER AWARE PROTOCOL ############
a=1;
bop=optimal_b(T,C,D,R,s);
Eop=energy_model2(C,D,R,s,bop);
```

```
while a >= 0
    n_tx3=n_tx3+1;

    for i=1:length(C)
        node_energy3(i)=node_energy3(i)-Eop(i);
        if node_energy3(i) <= 0
            fprintf('AWARE protocol:number of total tx was %d \n',n_tx3);
            a=-1;
        end
    end
end
%########################POWER AWARE PROTOCOL (BALANCING)#############
a=1;
while a >= 0

    %main also computes the tx of 1 packet over all the path
    [node,groups_balanced,not_balanced]=main(C,D,T,R,s,node_energy);
    n_tx=n_tx+1;


    for i=1:length(node)
        node_energy(i)=node(i).energy;
        C(i)=node(i).C;
        D(i)=node(i).D;
        b(i)=node(i).b;

        if node_energy(i) <= 0
            fprintf('node %d is down \n',node(i).index);
            fprintf('number of total tx was %d \n',n_tx);
            a=-1;
        end
    end

    if a==-1
         return
    end

    E=energy_model2(C,D,R,s,b);
    for i=1:rate
        n_tx=n_tx+1;
        for i=1:length(node)
            node_energy(i)=node_energy(i)-E(i);
            node(i).energy=node_energy(i);
            if node_energy(i) <= 0
                fprintf('node %d is down \n',node(i).index);
                fprintf('number of total tx was %d \n',n_tx);
                a=-1;
            end
        end

        if a==-1
            return
        end
    end
end
```

B.2 Tree and Exponential Distribution Simulator

```
function [node,n_tx,n_tx2,n_tx3,n_tx4,node_energy,node_energy2,node_energy3,↙
node_energy4]=sim_distr(C,D,T,R,s,node_energy,level,distr)
%Simulator for tree topologies or other kinds.
%INPUTS------------------------------
%C is a vector(row) that symbolizes the tx power
%D is a vector(row) that symbolizes the electronic power
%R is the transmission rate
%s is the packet size
%T total path latency
%node_energy is the energy of each node
%The consumption factor of each node is given by the parameter distr.
%Parameter level determines at what energy level we want to start to
%apply our protocol.
%OUTPUTS----------------------------
%n_tx is the nº of transmission for our protocol
%n_tx2 the transmissions for the Non-Power-aware
%n_tx3 the transmissions for the Power-aware
%n_tx4 the transmissions performed with the power-aware when a certain
%level lower than the max battery is specified
%-----------------------------------
%rate determines in our algorithm the up-date frequency
%of the modulation parameters of the whole path
b=0;
rate=100;
n_tx=0;
n_tx2=0;
n_tx3=0;
n_tx4=0;
node_energy2=node_energy;
node_energy3=node_energy;
node_energy4=node_energy;
coef=distr;
%#########################NON POWER AWARE PROTOCOL####################
a=1;
Emax=energy_model2(C,D,R,s,10);
while a >= 0
        n_tx2=n_tx2+1;

        for i=1:length(C)
            node_energy2(i)=node_energy2(i)-(coef(i).*Emax(i));
        end

        for i=1:length(C)
            if node_energy2(i) <= 0
                fprintf('NON-AWARE protocol:number of total tx was %d \n',n_tx2);
                a=-1;
            end
        end
end
%########################POWER AWARE PROTOCOL #############
a=1;
bop=optimal_b(T,C,D,R,s);
Eop=energy_model2(C,D,R,s,bop);
while a >= 0
    n_tx3=n_tx3+1;

    for i=1:length(C)
        node_energy3(i)=node_energy3(i)-(coef(i).*Eop(i));
    end
    for i=1:length(C)
        if node_energy3(i) <= 0
            fprintf('AWARE protocol:number of total tx was %d \n',n_tx3);
            a=-1;
        end
    end
end
%########################POWER AWARE PROTOCOL (BALANCING)#############
a=1;
bop=optimal_b(T,C,D,R,s);
Eop=energy_model2(C,D,R,s,bop);
while a >= 0
    n_tx4=n_tx4+1;

    for i=1:length(C)
        node_energy4(i)=node_energy4(i)-(coef(i).*Eop(i));
    end

    for i=1:length(C)
        if node_energy4(i) <= level
            fprintf('changing to balancing mode');
            a=-1;
        end
    end
end
for i=1:length(C)
    node_energy(i)=node_energy4(i);
end
a=1;
n_tx=n_tx4;
while a >= 0

    %main also computes the tx of 1 packet over all the path
    [node,groups_balanced,not_balanced]=main(C,D,T,R,s,node_energy);
    n_tx=n_tx+1;
    for i=1:length(node)
        node_energy(i)=node(i).energy;
        C(i)=node(i).C;
        D(i)=node(i).D;
        b(i)=node(i).b;
    end

    E=energy_model2(C,D,R,s,b);

    for i=1:length(C)
        node_energy(i)=node_energy(i)-((coef(i)-1).*E(i));
        node(i).energy=node_energy(i);
    end


    for i=1:length(node)

        if node_energy(i) <= 0
            fprintf('node %d is down \n',node(i).index);
            fprintf('number of total tx was %d \n',n_tx);
            a=-1;
        end
    end

    if a==-1
         return
    end
    for i=1:rate

        for i=1:length(C)
            node_energy(i)=node_energy(i)-(coef(i).*E(i));
            node(i).energy=node_energy(i);

        end

        n_tx=n_tx+1;

        for i=1:length(node)
            if node_energy(i) <= 0
                fprintf('node %d is down \n',node(i).index);
                fprintf('number of total tx was %d \n',n_tx);
                a=-1;
            end
        end

        if a==-1
            return
        end
    end
end
```