WARSAW UNIVERSITY OF TECHNOLOGY

Faculty of Electronics and Information Technology

The improvements of power management for clustered type large scope wireless sensor networks

by

Pedro de la Fuente Aragón

Supervised by

Daniel Paczesny, Ph.D.

June 2010

Table of contents

Index of Figures

Index of Tables

1 Introduction

1.1 Motivation

Sensors are a high developed technology integrated into very different areas like structures, machinery or the environment. Some of the potential benefits that they provide are: prevent catastrophic failures, enhance the job safety or conservation of natural resources. However, this sensor networks are typically wired networks and present high installation and maintenance costs, making more complicated their introduction and use in the daily life.

Wireless Sensor Networks (WSNs) are a new kind of communication network based in the use of new microelectronic devices called motes with sensing and data processing capabilities. This kind of networks can eliminate the installation and maintenance costs of typical sensor monitoring, in addition of its ease of installation and elimination of connectors. However, wireless devices contain battery constraints which limit the network lifetime. Due to the energy constrains, the deployment of large scope WSNs will require advanced techniques to maintain low node depletion and achieve adequate network lifetime and efficient operation.

The main goal of the current research is the analysis and the deployment of power management improvements based on the state-of-the-art of Media Access Control (MAC) and network protocols by means of simulation techniques. The implementation of power management improvements over large scope WSN will enhance the network lifetime without reducing the network features and capabilities.

During this chapter, a description with the fundamentals of Wireless Sensor Networks has been offered, as well as the benefits that WSN provide and the drawbacks that this kind of networks contains.

1.2 Description

The aim of this document is the research about the impact of some parameter modification within the mobility, MAC and network modules into the sensors' energy consumption. The improvements of the nodes' power management will be found through the study of the network behavior with different parameter values and how they will affect the energy consumption. Special importance will be present on how much influences the node mobility with particular speed values.

The differences in the network behavior will help to search different mechanisms to improve the power management and thus to reduce the energy consumption.

For that purpose, this project expects to built a realistic simulation of a WSN with the own constraints and required features of some WSN specific scenarios where energy consumption improvements could be obtained.

Initially, it is essential to insist not only on the capabilities of WSNs but also in their wide variety of applications and the benefits that provides their use.

The simulated network will have the common features which are present on the majority of the up-to-date WSN implementations. Furthermore, advanced features and

parameters will be analyzed in order to obtain an energy consumption improvement. Therefore, it is essential to analyze and study the state of the art of the most common WSN architectures and protocols.

After the analysis of the architectures and protocols present in the current WSNs, the most used simulation environments will be evaluated and the best simulation environment for the purpose of this project will be chosen and analyzed.

The main job will consist on the WSN scenario creation on the chosen simulation environment and the study of the network behavior after the variation of some critical parameters in order to obtain the values which provide the best results in terms of energy efficiency and network's functions.

1.2.1 What is a Wireless Sensor Network

According to the definition given in [1], *<<A wireless sensor network (WSN) consists of densely distributed nodes that support sensing, signal processing, embedded computing, and connectivity; sensors are logically linked by self-organizing means. WSN typically transmit information to collecting (monitoring) stations that aggregate some or all of the information. WSN have unique characteristics, such as, but not limited to, power constraints and limited battery life for the WNs, redundant data acquisition, low duty cycle, and, many-to-one flows.>>*. Although the development of this kind of networks was motivated by military applications, nowadays they are use in many different industrial and civilian application areas, including industrial process monitoring and control, healthcare applications or traffic control.

WSN are composed of a set of sensor nodes, called "motes", typically equipped with some sensors, a radio transceiver or other wireless communications device, a small microcontroller, and an energy source, usually a battery. Therefore, these devices make up a network with sensing, data processing and routing capabilities.

1.2.2 Benefits of Wireless Sensor Networks

To be knowledgeable about the benefits of WSNs, it is enough to be conscious of the wide variety of applications where WSN can be present. Typically, WSN applications involve some kind of monitoring, tracking, or controlling. Although wired sensor networks usually can develop the same function like WSNs, these last networks can be present in some applications where wired connections difficult the machinery function or are impossible to introduce, adding to the elimination of installation and maintenance costs.

Some of the numerous applications and the benefits that WSN bring are:

- Environmental Monitoring: watershed management, forest fire prediction or irrigation management. It helps to preserve and maintain the natural resources.
- Structural Health and Industrial Monitoring: machinery failure detection. It reduces the maintenance costs and prevents from catastrophic failures.
- Civil Structure Monitoring: health monitoring of large civil structures, like bridges or skyscrapers. It prevents from human catastrophes.
- Medical Health-care: telemedicine, remote health monitoring. Allows doctors in remote and rural areas to consult with specialists in urban areas, remote handling of medical equipment (tele-surgery), etc.

1.2.3 Drawbacks of Wireless Sensor Networks

Besides WSN offer several additionally advantages to wired sensor networks, they impose some important constraints, which will affect directly to the network's and devices' design. Into the Section 2.3 - Design factors and requirements, the main constraints and design requirements are described. Some of the most significant constraints are:

- Power consumption: this constraint affects directly into the nodes' lifetime. With energy-aware and transmitting power adjusting capacity protocols, the energy consumption can be highly reduced, and thus increased the network lifetime.
- Self-configuration capability and good scalability: this issue can be solved by choosing and implementing the suitable network protocol.
- Fault tolerance: if all the devices process the same signal (temperature, humidity, etc.), the network will offer replication in a native manner. If the devices do not develop the same function, the device replication can solve the fault tolerance problem, and this solution shouldn't affect the scalability due to the nature of the network.

1.3 Objectives

1.3.1 Main objectives

The main objective of the current project is the search, the development and the implementation of a power management improvement through the adaptation of existing MAC and routing protocols for specific large scope WSN scenarios by means of simulations of the network behavior.

1.3.2 Secondary objectives

Diverse secondary objectives are pursued during the development of the current project:

- Objectives into the election of the protocols. The protocols that should implement the network of the current project ought to fulfill specific requirements. The MAC protocol should provide a restrained energy consumption, and the routing protocol should present a cluster-type organization and provide a balanced overall energy consumption, i.e., distribute the energy consumption along the nodes in the network.
- Objectives into the simulation implementation. The simulation environment should fulfill some requirements like WSN network architecture and features, portability, open source development and a good and friendly user interface. Furthermore, the network simulation implementation should make use of efficiency, independence between modules and reusability principles.

1.4 Description of the document

In this document, ten chapters have been presented which presents the performed work.

- In the chapter 2, the communication architecture, the architecture of the protocol stack and the featured design factors and requirement of WSNs are detailed.

Furthermore, the state of the art of the MAC and network WSN protocols is analyzed.

- In the chapter 3, the requirements of a WSN simulation environment is analyzed. Moreover, the most common network simulation environments are analyzed.
- In the chapter 4, the evaluation of the chosen WSN routing protocols is introduced through their operation description and their implementation into the simulation environment.
- In the chapter 5, the network simulation scenarios are depicted, as well as the implementation of the network architecture, the physical environment and the internal node architecture.
- In the chapter 6, the radio model adopted for the simulation tests and the parameters selected are described. Furthermore, a priory analysis of the results is given.
- In the chapter 7, the different simulation tests made over the network are detailed, and the results obtained after the simulation execution of the different simulation scenarios are analyzed.
- In the chapter 8, the obtained conclusions during the development of the whole project are detailed.
- In the chapter 9, the possible improvements that could be implemented in future work are explained.
- In the chapter 10, the external references consulted during the project development are listed.

2 State of the art

2.1 Introduction

WSN appear as a new and revolutionary way of communication. This communication method provides numerous advantages but also contains several constraints. In this section, the main entities of WSN communication architecture, the network requirements that this architecture should provide and the design factors that WSN constraints add in their operation are analyzed. After that, the three layers of WSN networks that provide the communication bases are reviewed. In MAC and routing protocols subsections, it is important to examine the variety of protocols which implement the functionality of these MAC and routing layers and provide different features to the network depending on a specific kind of network or the network behavior that it is expected to obtain. In the end, some conclusions of this review are extracted.

2.2 The communication architecture

A Wireless Sensor Network is composed of a set of numerous sensors with sensing, wireless communication and computation capabilities. These sensors are scattered in an unattended environment and located away from the user.

The main entities which compose the WSN architecture [2] are:

- Sensors which make up the network: its function is based on taking local measures through a discrete system, creating a wireless network in an unattended environment, gathering data and sending them to the final user through the base station.
- Base station or gateway node: it is located near the sensor field. The data or information gathered by the sensor field is sent to the base station through a multihop infraestructureless architecture, which communicates with the user via Internet or satellite communication.
- User: it is the entity interested in obtaining the information about a specific phenomenon by means of measuring or monitoring the environment.

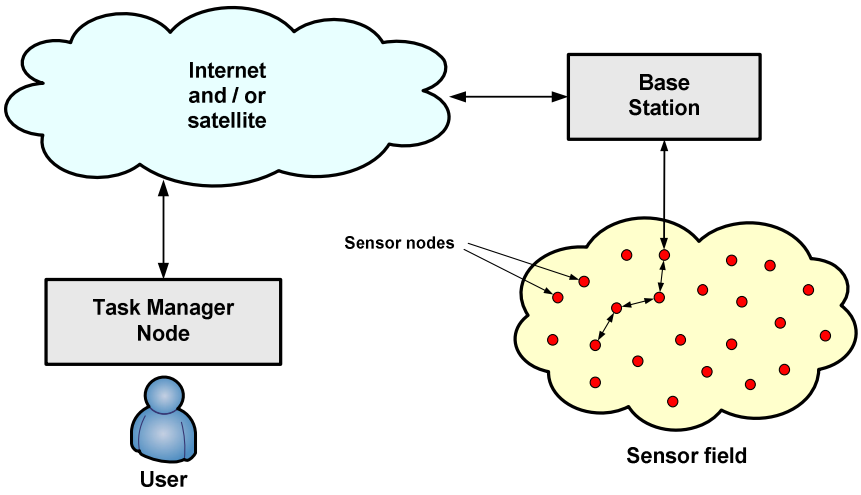

Figure 1. Wireless Sensor Network architecture

2.3 Design factors and requirements

Wireless Sensor Networks are composed of small devices with wireless communication, sensing and computation capabilities. These devices consist of only a small memory, a short range radio and a battery. Consequently, the constraints which impose these devices together with the characteristics, which are typical of this kind of networks, establish some guidelines for the protocols or algorithms design in WSNs. Detailed below are the main requirements [3] of WSNs:

- Reliability and/or fault tolerance: capacity of the WSN to operate without any interruption.
- Density and scalability: the density affects the network coverage degree, and the size of the network affects the reliability and data processing algorithms.
- Network topology: it concerns directly different characteristics as network latency and robustness. It determines the complexity of routing protocols.
- Power consumption: the sensors' life time depends directly on the battery life time. Therefore, current researches are focused on protocols and algorithms design which are power-aware and consider the importance of minimizing the power consumption.
- Data aggregation and fusion: they have the goal of reducing the data size with computation methods in order to decrement the network traffic and consequently the network congestion.
- Transmission media: radio, infrared, optical, etc.
- Quality of Service (QoS): in some applications, the data time constraints can be critical for the correct operation of the WSN, meanwhile in other applications becomes more important the life time.
- Hardware constraints: network nodes are usually composed by two subsystems: sensor system and ADC (Analog to Digital Converter) system. Besides the sensing, computation, transmission and power unit, they can contain other components as position/location finding systems or power generator systems.
- Self-configuration: it is an essential issue on WSN owing to two different factors: the possibility of fault or addition of new nodes to the network and the network operation capacity in an unattended way.
- Other requirements: security, network dynamics, connectivity, etc.

2.4 The architecture of the protocol stack

The architecture of the protocol stack [4] of WSNs differs slightly from the Open System Interconnection Reference (OSI) model. This protocol stack integrates other features as power aware or data with network protocols (data aggregation/fusion). Meanwhile OSI model presents seven layers; WSNs' protocol stack reduces the model to five levels and incorporates two planes. Detailed below are the different layers and planes of the WSN's architecture:

- Physical layer: it provides robust modulation, transmission and receiving techniques.
- Data link layer: it establishes the functional and procedural means to transfer data between network entities and error detection techniques.
- Network layer: it is in charge of routing the data supplied by the transport layer.
- Transport layer: it establishes a flow data if the WSN application needs it.
- Application layer: it depends on the phenomenon of interest and the sensing tasks.
- Power management plane: it manages the power consumption of the tree main tasks of a sensor node: sensing, computation and communication.
- Mobility management plane: it registers the movement and the location of all the nodes as a network control primitive.
- Task management plane: it manages and schedules the sensing and detecting tasks in order to obtain balanced power consumption.

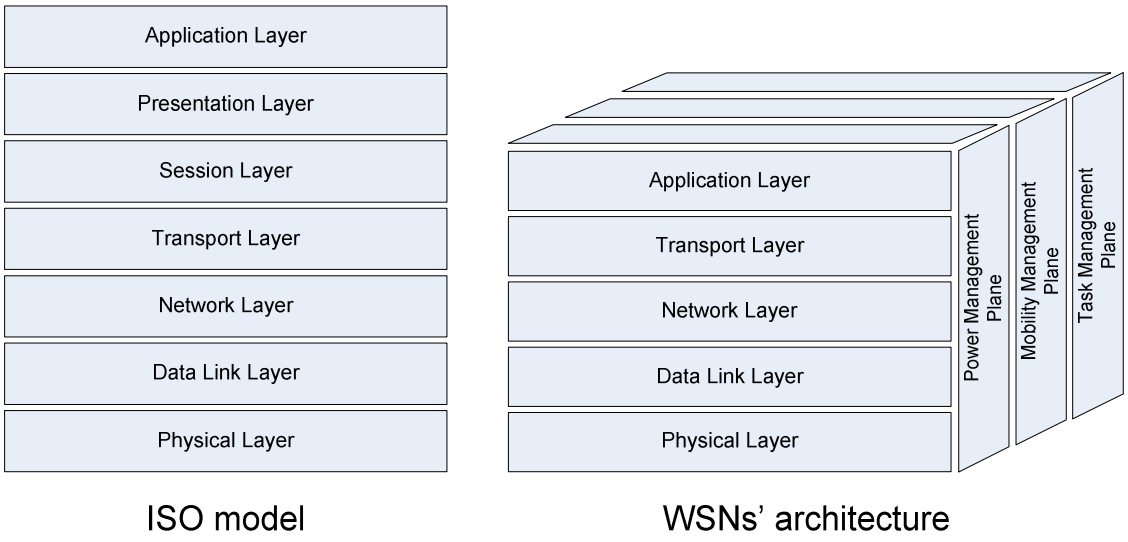

Figure 2. Comparison between OSI model and WSN's stack protocol architecture

2.5 Wireless Sensor Network protocols

2.5.1 The importance of the physical layer on Wireless Sensor Network protocols

The physical layer, as detailed in [2], takes charge of the frequency selection, carrier frequency generation, signal detection, modulation and demodulation of digital data and data encryption; and this task is carried out by transceivers. WSN transceivers show a common structure on Radio Frequency (RF) front-end, as illustrated on Figure 3, and the baseband part:

- The **RF front-end** performs analog signal processing in the actual radio frequency band, where the Power Amplifier (PA) amplifies signals from the baseband part, the Low Noise Amplifier (LNA) amplifies incoming signals, and other elements like oscillators and mixers are used for frequency conversion.
- The **baseband processor** performs all signal processing in the digital domain and communicates with node's processor or other circuitry.

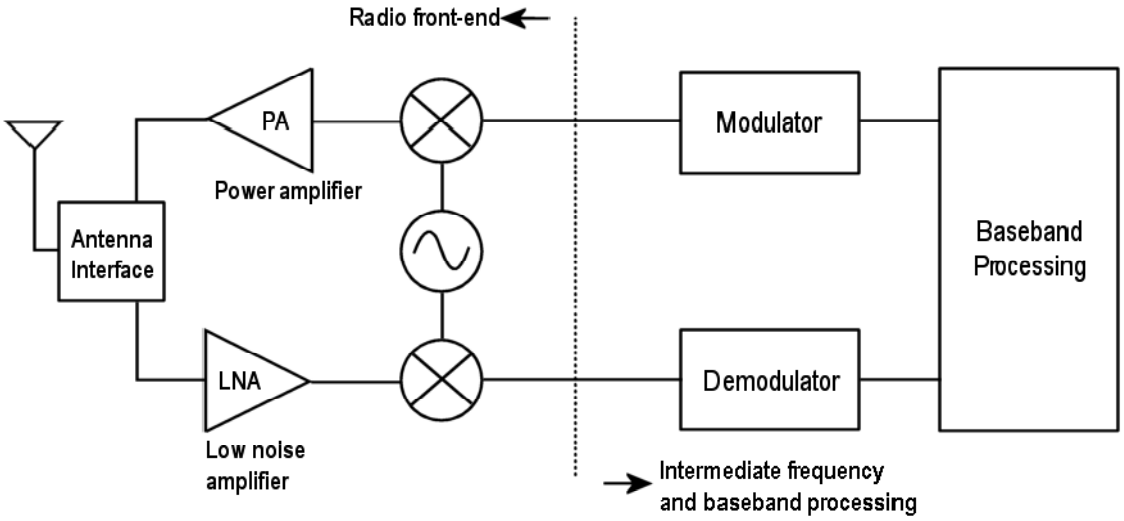

Figure 3. RF front-end and baseband processor

The desires of the current researches about the physical layer on WSNs are focused on the search of cheap, effective and simple modulation schemes and transceiver architectures which perform the required task. Thus, the main issue is how to transmit as energy efficiently as possible, taking into account all related costs (overhead, possible retransmissions etc.), considering scattering, shadowing, reflection, diffraction, multipath and fading effects typical of wireless transmissions. To do this, it's worth keeping in mind the problems which appear in every digital communication over wireless channels, as well as the problems and constraints which are added by the specific WSNs requirements.

Some aspects to consider in wireless communication are:

- Frequency allocation: it is very important to choose carefully the carrier frequency in a radio frequency (RF)-based system because it determines the propagation characteristics. The range of radio frequencies is subject to regulation to avoid unwanted interferences between users and systems. But besides the special licenses for reserved bands, there are also lisencefree bands (Industrial, Scientific and Medical (ISM) bands) although these ISM bands adds

the problems of living with interference created by other systems (as for example IEEE 802.11 and Bluetooth systems with the 2.4 GHz band).

- Modulation/demodulation scheme: this is a very important point. To obtain the best results, several factors have to be balanced: the required and desirable data rate and symbol rate, the implementation complexity and the relationship between radiated power and target Bit Error Rate (BER). In order to maximize the time a transceiver can spend in sleep mode, the transmit times should be minimized.
- Wave propagation effects and noise: waveforms transmitted over wireless channels are subject to several phenomena that all distort the original transmitted waveform at the receiver, like reflection or diffraction. This distortion introduces uncertainty at the receiver about the originally modulated data, and can result in bit errors.

It is important to analyze the different kind of modulation schemes in order to select the minimum consumption solution. In WSN, simple modulation techniques are selected because of their easiness of implementation, robustness and low power consumption. The common used modulation schemes are:

- Amplitude-shift keying (ASK): form of modulation that represents digital data as variations in the amplitude of a carrier wave.
- Frequency-shift keying (FSK): frequency modulation scheme in which digital information is transmitted through discrete frequency changes of a carrier wave.
- Binary phase-shift keying (BPSK): digital modulation scheme that conveys data by changing, or modulating, two phase of a carrier wave separated by 180º.
- Quadrature amplitude modulation (QAM): a combination of both phase-shift keying (PSK) and amplitude-shift keying (ASK).

With the aim of reducing the transmit time of the radio, an m-ary modulation (for example, 4-ASK, 4-PSK or 16-QAM) and *dynamic modulation scaling* (modulation scheme adaptation for different situations) can be used. This modulation sends multiple bits per symbol, i.e., it obtains high data rates at low symbol rates. However, an m-ary modulation will increase the circuit complexity and power consumption of the radio. Furthermore, with m-ary, efficiency of the power amplifier is also reduced. Therefore, the optimal decision will balance properly the modulation scheme and other measures to increase transmission robustness.

After the analysis of the most important aspects to consider in wireless communications, it is appropriate to know the most crucial points concerning physical layer design in wireless sensor networks, as it's possible to see on the next list:

- Low power consumption.
- Small transmit power and small transmission range as a result of the previous characteristic.
- Low duty cycle in order to save energy by means of switching off most hardware or operating in a low-power standby mode most of the time.
- Low implementation complexity and costs.

2.5.2 MAC protocols

2.5.2.1 Causes of energy waste concerning the MAC layer

Energy waste's main reasons are *collisions*, *overhearing*, *control packet overhead*, *idle listening* and *overemitting*. *Collisions* consist on the reception of more than one packet at the same time with the result of discarding and packet retransmission. *Overhearing* occurs when a node receives packets destined to other nodes. The *control packet overhead* or the number of control packets should be minimized as far as possible in a data transmission. *Idle listening* is produced when a node listens to an idle channel to receive possible traffic. And *overemitting*, which is caused by the transmission of a message when the destination node is not ready. A correctly-designed MAC protocol should avoid these facts in order to obtain the best performance and the minimum energy consumption.

The following MAC protocols will be analyzed in order to obtain a general operation idea and extract their advantages and drawbacks:

- Sensor-MAC (S-MAC): protocol based on locally managed synchronizations.
- Timeout-MAC (T-MAC): improvement of S-MAC with variable listen periods.
- Dynamic Sensor-MAC (DSMAC): variation of S-MAC with dynamic duty cycle.
- Berkeley MAC (B-MAC): protocol based on different check intervals corresponding with different listening modes.
- Wireless Sensor MAC (WiseMAC): protocol based in a preamble sampling technique.
- DMAC: this protocol makes use of a *convergecast* communication pattern.
- Traffic-Adaptive MAC (TRAMA): protocol based in a distributed election algorithm.
- SIFT: protocol used on *event-driven sensor network environments* and based on a data priority schema.
- CSMA: this protocol verifies the absence of other traffic before transmitting on a shared transmission medium.

2.5.2.2 Description of proposed MAC protocols

S-MAC

The basic idea of Sensor-MAC [5] protocol consists on locally managed synchronizations and periodic sleep listen schedules based on these synchronizations. Nodes sleep and wake up periodically introducing the term of *duty cycle*. This protocol shows a drawback: when two neighbor nodes reside in two different virtual clusters which set up a common sleep schedule, they wake up at listen periods of both clusters.

Synchronization is required by this exchange schedule schema, which is provided through SYNC packet broadcasts within a virtual cluster. Collision avoidance is achieved by a carrier sense, RTS/CTS packet exchanges prevent from the *hidden node* problem, and *adaptative listening* can be used in order to reduce the sleep relay and thus the overall latency.

The advantages of this protocol consist on its implementation simplicity and its energy consumption decrease through sleep schedules. But the disadvantages are the collision probability with broadcast data packets because of the lack of a RTS/CTS schema, the efficiency loss with its constant and predefined sleep and listen periods, overhearing and idle listening problems.

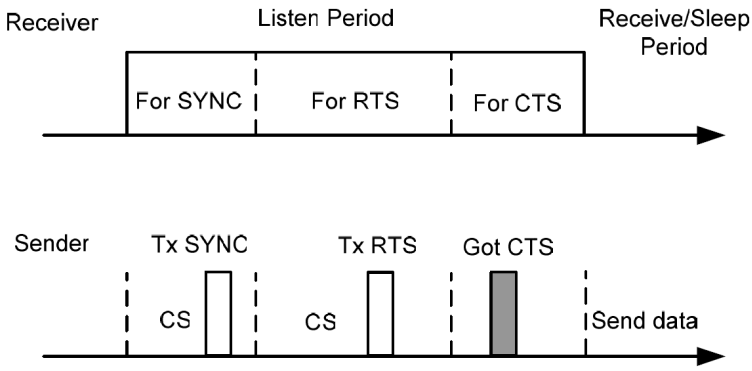

Figure 4. S-MAC Messaging Scenario [5]

T-MAC

Timeout-MAC [6] protocol is an improvement of S-MAC protocol which tries to provide higher energy saving under variable traffic load through variable listen periods. In T-MAC, listen period finishes when a node doesn't have messages to send or receive in order to save energy. After the message exchange, the node waits an activation event for a time threshold.

Although T-MAC shows better results than S-MAC, it breaks the listen period's synchronization and, because of this and other reasons, T-MAC protocol suffers the "early sleep" problem, where a node C within the same coverage area than a node B can't send a message to a node D because the node B is receiving a message from a node A and the node D doesn't detect any sign of activity and switches to sleep mode.

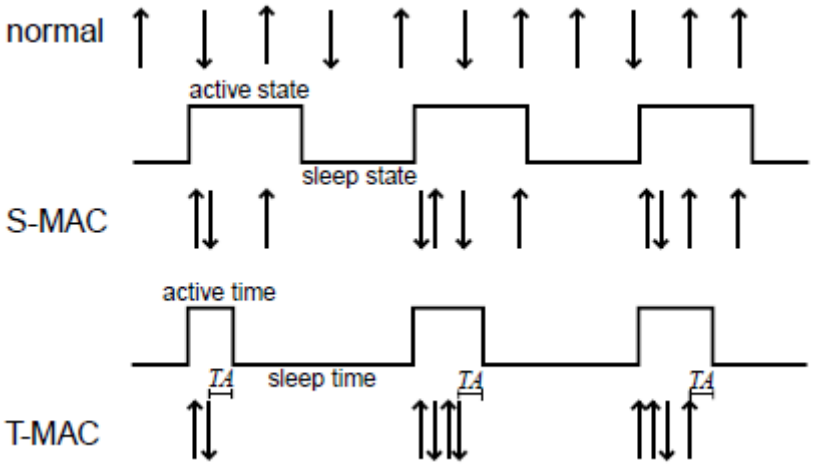

Figure 5. Comparison between S-MAC and T-MAC schemes, where the arrows indicate transmitted and received messages

DSMAC

Dynamic Sensor-MAC [7], also called DSMAC, is a protocol which adds *dynamic duty cycle* to S-MAC and attempts to decrease the latency for delay-sensitive applications. In this protocol all nodes start with the same duty cycle, and when a node realizes that average one-hop latency is high, it decides to shorten its sleep time and announces it within SYNC period. As a consequence, after a sender node receives this signal, it checks its queue for packets destined to that receiver node and decides to double its duty cycle when its battery level is above a specified threshold. In this manner, DSMAC improves the latency obtained with S-MAC and shows better average energy consumption.

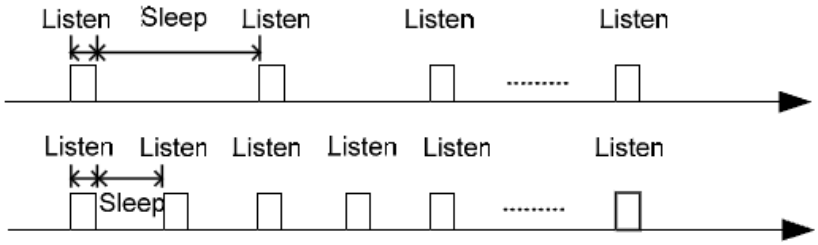

Figure 6. DSMAC duty cycle doubling [7]

B-MAC

Berkeley MAC [8] protocol comes from the University of California, Berkeley, and it achieves to decrease the idle listening. B-MAC proposes that each node must sleep periodically to check the channel occupation; if a node detects activity it remains in listening mode, otherwise it switches to sleeping mode. B-MAC defines eight different *check intervals* or time intervals between wake-up periods, and each one corresponds with a different *listening mode*. In order to assure packet delivery, packets are sent with a preamble whose length transmission is longer than the check interval.

The advantages of B-MAC are the simplicity of network configuration, ease of tuning, no necessity of explicit sync packets and don't use of RTS/CTS/ACK if not necessary.

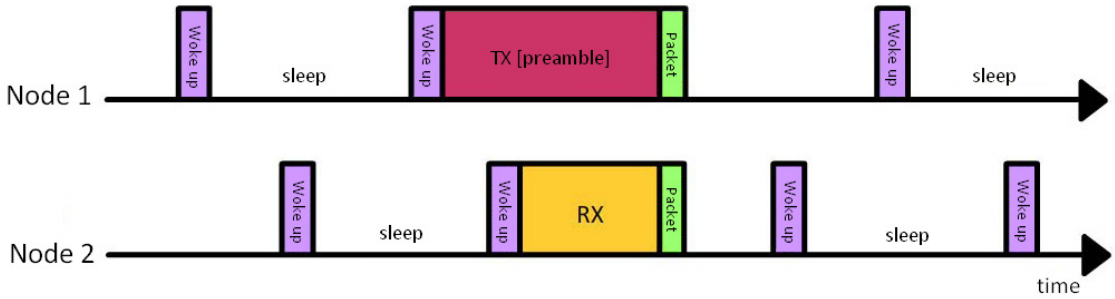

Figure 7. B-MAC concepts

WiseMAC

The Wireless Sensor MAC [9] protocol introduces a new communication schema with a data channel access by spatial TDMA and gives access to the control channel by CSMA. This protocol is based in a preamble sampling technique, where each data packet is preceded by a preamble in order to alert the receiver node. All network nodes sample with a common media period, but using independent *relative schedule offsets*.

They initialize the preamble with the same sampling period's length. During the protocol's use, after waking and sampling the media when a node reaches an it's occupied, stays hearing until receives a packet or finds free the media. This protocol has overemitting problems when after the preamble, the receiver is not available. Also, with the aim of reducing the energy consumption, WiseMAC offers a dynamic length definition preamble method which requires *sleep schedules* learning neighbor nodes, achieving to minimize the receiver nodes' radio working time. On the contrary, the difficult of broadcast communication due to the decentralized duty cycle planning and the hidden terminal problem apparition are the main inconvenients.

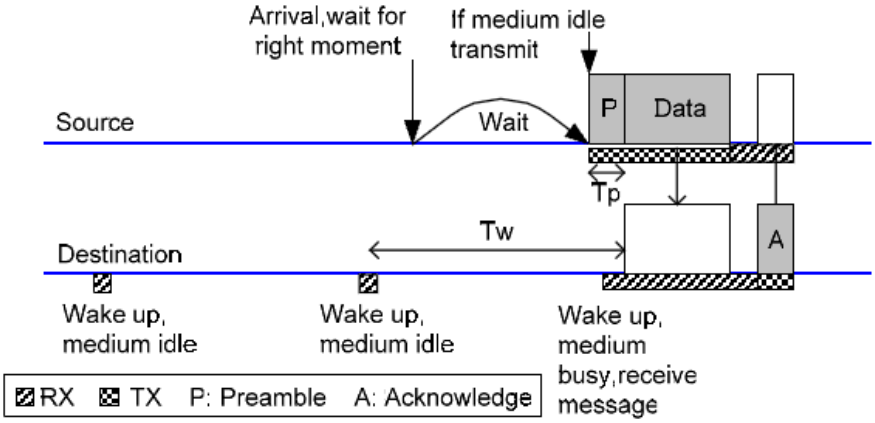

Figure 8. WiseMAC operation

DMAC

DMAC's [10] main objective consists on obtaining a very low latency by means of a *energy-efficient* operation. This protocol makes use of a *convergecast* communication pattern, very applied on WSNs, where unidirectional paths from the possible sources to the base station can be represented with *data gathering trees*. DMAC can be identified as an improvement of slotted Aloha protocol, where slots are assigned to sets of nodes based on a data gathering tree similar as showed on Figure 9. In this manner, during a node reception period, all its son nodes have also the same transmission period and they compete for the media. Thus, this protocol provides low latency by assigning contiguous slots to the consecutive nodes along the transmission path.

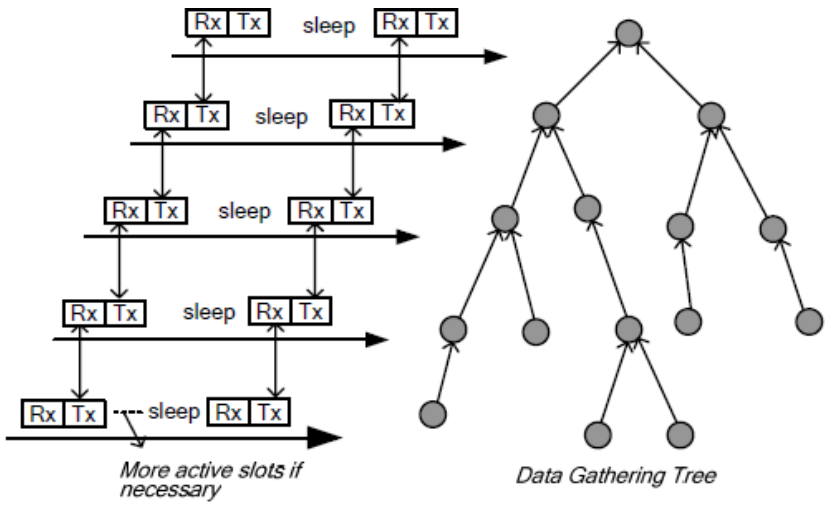

Figure 9. Data gathering tree and implementation over DSMAC [10]

One of the best features of DMAC is very good latency in comparison with other sleep/listen period assignment methods. Hence, this protocol becomes a very important candidate in time-constrained applications. On the contrary, this protocol doesn't use *collision avoidance*. For this reason, when a considerable number of nodes on the same level try to send data to the same node, collisions will happen.

TRAMA

Traffic-Adaptive MAC [11] protocol is similar to Node Activation Multiple Access (NAMA) protocol which is operation is described in [11], but this increases the use of TDMA as an energy-efficient mode. In TRAMA protocol a distributed election algorithm is used in order to select a sender inside a two-hop neighborhood. By means of this mechanism, the hidden terminal problem is eliminated and nodes inside the one-hop neighborhood guarantee no collision packets will be received. In this registry, time is divided in two different transmission periods: random-access periods, where two-hop topology information through contention-based channel access, and scheduled-access. In these last ones, slots which will be used by nodes are announced by a schedule packet and the bitmap message scheduled receivers.

This protocol achieves important advantages: a sleeping mode time percentage increase and a collision probability decrement in comparison with CSMA based protocols. Even so, TRAMA *duty cycle* is at least of 12.5%, a considerable high value.

SIFT

SIFT [12] is a MAC protocol for WSN whose operation differs from the above described protocols. This protocol is used on *event-driven sensor network environments*. Its main idea consists on the next fact: when an event is sensed, the first R reports of N potential reports composes the most important communication part, and this part must be delivered with the minimum latency. SIFT uses a non-uniform probability distribution function. This function helps to the slot acquisition within the slotted contention window: if nodes don't transmit on the first window slot, all nodes increment exponentially its transmission probability on the next slot considering limited the number of competitors.

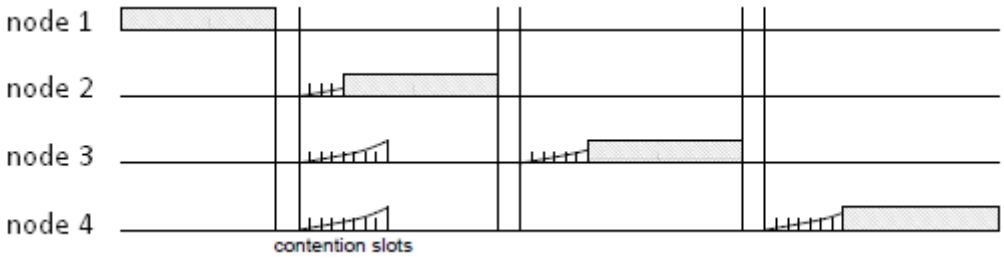

Figure 10. A timeline of four nodes running SIFT protocol, where shaded bars indicate packet transmission times and node's contention window are shown

This protocol reaches very low latency through a power consumption increment. This parameter can be set properly to the environment requirements. Thus, it could be possible to obtain a power consumption decrement losing some features as low latency when network life time is the main objective. As disadvantages, time on idle listening is increased due to the nodes must listen all the slots before its sending, as well an overhearing increment.

CSMA

In Carrier Sense Multiple Access (CSMA) [13], the nodes verify the absence of other traffic before transmitting on a shared transmission medium. Two versions of CSMA exist: non-persistent CSMA and p-persistent CSMA. In non-persistent CSMA, a backoff is performed before attempting to transmit if the sensed channel is busy, and the transmission is carried out immediately if the device senses no activity on the channel. In p-persistent CSMA, a node continues sensing the channel if it detects activity instead of delaying and checking again later. When the device senses no activity on the channel, it transmits a message with probability p and delays the transmission with probability $1-p$.

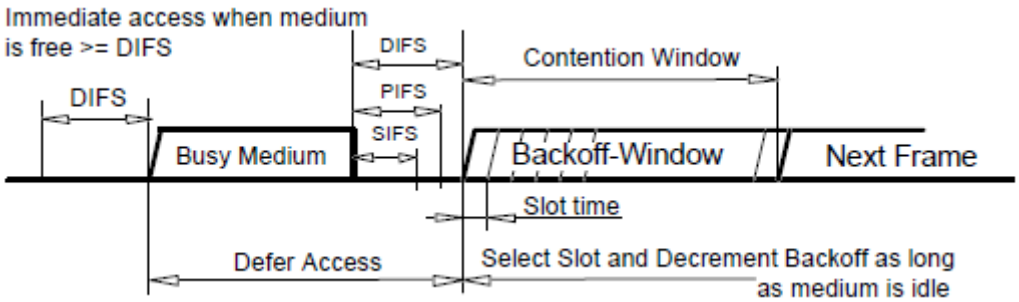

Figure 11: CSMA operation

The channel access times and backoff delays showed on Figure 11 consist of continuous values for unslotted CSMA or discrete time values for slotted CSMA. These parameters are explained below:

- SIFS: the minimum Inter Frame Space. It is used to separate transmissions belonging to a single dialog (e.g. Fragment-ACK).
- PIFS: it is used by the Access Point to gain access to the medium before any other station. The value of PIFS is SIFS plus a Slot time. Not important in WSN operation.
- DIFS: it is the Inter Frame Space used for a station willing to start a new transmission, which is calculated as PIFS plus one Slot time.
- Slot time: it is defined in such a way that a station will always be capable of determining if other station has accessed the medium at the beginning of the previous slot. This reduces the collision probability by half.

Backoff is method to resolve contention between different stations willing to access the medium. The method requires each station to choose a Random Number (n) between 0 and a given number (*Contention Window* value), and wait for this number of Slots before accessing the medium, always checking whether a different station has accessed the medium before.

The benefit of CSMA/CA techniques in sensor networks depends on the traffic conditions, wireless channel characteristics, and network topology, so in some cases it may prove beneficial and in others an unnecessary overhead.

2.5.3 Routing protocols

This section presents the classification of WSN routing protocols. Routing protocols can be divided into three groups depending on the network structure: *flat-based* routing, *hierarchical-based* routing, and *location-based* routing. Furthermore, these same protocols can also be divided into five different groups depending on the protocol operation: *multipath-based*, *query-based*, *negotiation-based*, *QoS-based*, and *coherent-based* routing. In addition, routing protocols can be divided into *proactive*, *reactive* and *hybrid* protocols depending on how the source finds a route to the destination. In proactive protocols, all routes are computed before they are needed. In reactive protocols, on the contrary, routes are computed on demand. Hybrid protocols use a combination of these two techniques. Figure 12 shows the classification of WSN routing protocols. Detailed below are the different routing paradigms.

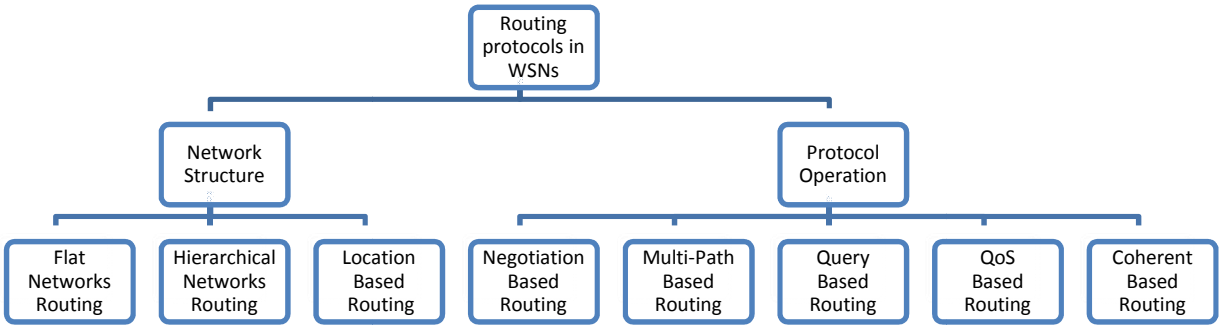

Figure 12. Classification of routing protocols in WSNs

2.5.3.1 Network Structure Based Protocols

The network structure adopts an important role in the operation of routing protocols in WSN. Next are exposed the different subgroups according to the diverse network structures.

Flat routing

In multihop flat routing protocols, all nodes plays the same role and collaborate together to perform the sensing task. They make use of a data centric routing scheme, where the BS sends queries to certain regions and waits for data from the sensors located in the selected regions. This schema has been adopted due to the large number of nodes, which makes impossible to assign a global identifier to each node. In flat routing, data is requested through queries. Hence, attribute-based naming is necessary to specify the properties of data. In flat routing group, we can find a huge variety of protocols:

- *Flooding*, where data is sent to all neighbors; and *Gossiping* [14], based on Flooding, where data is forwarded to one randomly selected neighbor. They are the simplest protocols, but contain important drawbacks (as implosion, overlapping and resource blindness) and they work in a very inefficient way.
- *Sensor Protocols for Information via Negotiation (SPIN)* [15]: it is based on operating efficiently by sending meta-data and being aware and answering in view of energy resource changes. This protocol solves the problems of above protocols through data negotiation and resource adaptive algorithms. SPIN has some drawbacks, as scalability problems, and delivery not guaranteed.

- *Directed Diffusion* [16]: it consists basically of naming, interests and gradients, data propagation along the interest's gradient path, and paths' reinforcement. This protocol saves energy by selecting good paths and performing data aggregation and caching. Otherwise, data aggregation requires synchronization techniques and increments and recording information when overhead appears. There are several protocols based on Directed Diffusion:
    - *Rumor routing* [17], whose key idea is to route the queries to the nodes that have observed a particular event rather than flooding the entire network, and if flooding is needed, it employs long-lived packets called agents. Rumor routing achieves energy savings but it has scalability problems.
    - *Gradient-Based Routing (GBR)* [18], where nodes calculate a parameter called the height of the node memorizing the number of hops when the interest is diffused and forward packets on links with the largest gradient or difference between neighbor nodes' height.
    - *Information-driven sensor querying (IDSQ)* and *Constrained anisotropic diffusion routing (CADR* [19]*)*, where queries are diffused in an isotropic fashion and reaching nearest neighbors first. They query sensors and route data by maximizing information gain and minimizing latency.
    - *Energy Aware Routing* [20], which maintains a set of paths instead of or enforcing one optimal path at higher rates. It employs a kind of probability whose value depends on how low is the energy consumption of each path. It achieves to increment network lifetime.
- *Minimum Cost Forwarding Algorithm (MCFA)* [21]: in this protocol, each node maintains the least cost estimate from itself to the base-station. Thus, nodes only re-broadcast messages to their neighbors when they check that they are in the least cost path between the source and the base-station.
- *COUGAR* [22] and *ACQUIRE* [23]: these protocols view the network as a huge distributed database system. COUGAR uses declarative queries, whereas ACQUIRE can divide complex queries into several sub queries. They show energy efficiency in situations when the generated data is huge.

Hierarchical routing

In WSN, the concept of hierarchical or cluster based routing is utilized to perform energy efficient routing. In addition, this mechanism provides good scalability and efficient communication. Applying hierarchical architecture to WSN, higher energy nodes can be used to process and send the information while low energy nodes can be used to perform the sensing in the proximity of the target. Thus, cluster heads can perform data aggregation and fusion in order to decrease the number of transmitted messages to the BS and increment the network lifetime. However, in WSN, routing techniques are not only focused on routing, but also on "who and when to send or process/aggregate" the information, channel allocation, etc. Detailed below are several proposals of hierarchical routing protocols:

- *Low Energy Adaptive Clustering Hierarchy (LEACH* [24]*)*: is a self-organizing protocol that uses randomized rotation of cluster-heads to evenly distribute the energy load among the sensor nodes in the network. Although LEACH provides some added features like localized coordination and control, it can't be applied to time-constrained applications and also presents the "hot-spot" [25] problem.
- *Power-Efficient Gathering in Sensor Information Systems (PEGASIS)* [26]: based on LEACH, PEGASIS is a power efficient algorithm. In this protocol, each node can take turn of being a leader of the chain, where the chain is constructed using greedy algorithms that are deployed by the sensor nodes. PEGASIS outperforms LEACH in several aspects but it presents the same problems than LEACH and also doesn't scale.
- *Threshold-sensitive Energy Efficient Protocols (TEEN* [27] *and APTEEN* [28]*)*: these two protocols, proposed for time-critical applications, are LEACH based with multi-level head clusters. They are based on two values: a hard threshold, with the threshold sense value, and a soft threshold, a small value that triggers nodes to transmit. This method reduces the number of transmissions and these values can be tuned to increment the accuracy. Both two protocols outperform LEACH, but multi-level clusters and threshold-based functions increment their complexity.
- Other protocols: *Small Minimum Energy Communication Network (MECN)* [29], which computes an energy-efficient subnetwork by utilizing low power GPS; *Sensor Aggregates Routing*, whose objective is to collectively monitor target activity in a certain environment; *Self Organizing Protocol (SOP)*, where heterogeneous sensor architecture with mobile or stationary nodes is supported; *Virtual Grid Architecture routing (VGA)*, which uses a GPS-free approach to build clusters that are fixed, equal, adjacent, and non-overlapping with symmetric shapes; *Hierarchical Power-aware Routing (HPAR)*, which divides the network into groups of sensors where messages are routed along the path which has the maximum over all the minimum of the remaining power (max-min path); and *Two-Tier Data Dissemination (TTDD)*, which provides data delivery to multiple mobile base-stations.

As it is shown, there are many differences between flat and routing protocols. In Table 1 extracted from [30] these two approaches are compared.

Table 1. Hierarchical vs. flat topologies routing

Hierarchical routing	**Flat routing**
Reservation-based scheduling	Contention-based scheduling
Collisions avoided	Collision overhead present
Reduced duty cycle due to periodic sleeping	Variable duty cycle by controlling sleep time of nodes
Data aggregation by cluster head	Node on multihop path aggregates incoming data from neighbors
Simple but non-optimal routing	Routing can be made optimal but with an added complexity
Requires global and local synchronization	Links formed on the fly without synchronization

Hierarchical routing	Flat routing
Overhead of cluster formation throughout the network	Routes formed only in regions that have data for transmission
Lower latency as multiple hops network formed by cluster heads always available	Latency in waking up intermediate nodes and setting up the multipath
Energy dissipation is uniform	Energy dissipation depends on traffic patterns
Energy dissipation cannot be controlled	Energy dissipation adapts to traffic pattern
Fair channel allocation	Fairness not guaranteed

Location based routing

Location based routing shows a different schema where nodes are by means of their location. In this kind of routing, the location of nodes can be obtained directly if nodes are equipped with a low power GPS or their relative position can be deduced by means of two facts: the distance between nodes estimated on the basis of incoming signal strengths, and relative coordinates of neighbors obtained by exchanging information between them. Energy savings can be obtained switching nodes to sleep mode when there is no activity or having as many sleeping nodes in the network as possible. Next are reviewed several location-based routing protocols:

- *Geographic Adaptive Fidelity (GAF)* [31]: GAF divides the network area into fixed zones and forms a virtual grid. All nodes in each zone elect one sensor node responsible in its zone for monitoring and reporting data to the BS. This node will stay awake for a certain period of time and the rest of nodes will go to sleep. Thus, GAF conserves energy by turning off unnecessary nodes in the network without affecting the level of routing fidelity.
- *Geographic and Energy Aware Routing (GEAR)* [32]: GEAR is a recursive data dissemination protocol. This protocol disseminates queries to appropriate regions whose data include geographic attributes. It achieves energy saving by sending the interest to certain regions rather than the whole network. On the contrary, GEAR is not scalable and does not support data diffusion.
- *MFR, DIR, and GEDIR* [33]: these three protocols employ different mechanisms but they almost always obtain the same path to the destination. In MFR, the dot product of Euclidean distance between destination and neighbor node and Euclidean distance between destination an source node (i.e., $\overline{DA} \cdot \overline{DS}$) is minimized. DIR method chooses the neighbor with the minimum angular distance between from the imaginary line joining the current node and the destination. In GEDIR, packets are transmitted to the neighbor of the current vertex whose distance to the destination is minimized.
- *SPAN* [34]: this protocol selects some nodes based on their positions that will act as coordinators and will form a network backbone used to forward messages. A node is designed as coordinator when three hop reachability between nodes is not accomplished, i.e., two nodes cannot reach each other directly or via one or two coordinators.

2.5.3.2 Protocol Operation Based Protocols

In this subsection, some routing methods with different features and functionalities are described. Some of these methods may refer to some concrete protocol which deploys the main feature in their category.

Multipath routing protocols

Multipath routing mechanisms uses multiple paths in order to increase the fault tolerance of the network. These maintained alternative paths entail an increment of energy consumption and traffic overhead, but helps to increase the network reliability.

Diverse researches have provided different ideas. One proposal advocates to create paths with the largest residual energy, change the path whenever a better path is discovered and switch the primary path to the backup path when the energy of the primary path falls to lower levels than the backup path. Another proposal suggests us to use a set of sub-optimal paths with the less energy consumption. This method aims to increase the network lifetime by choosing the paths by means a certain probability depending on the lower minimum energy consumption of each path. Directed diffusion is also a good protocol for robust multipath routing and delivery.

Query based routing

In query based routing, base station or destination nodes propagate a query through the network. These queries usually use natural or high-level query languages. All nodes have tables with sensing task queries that they receive and nodes having the data associated to this query sends the data back to the node which made the query.

Several protocols use query based routing. For instance, in Directed Diffusion, interest messages are propagated through the network and gradient paths are set up and. When the source has data for the interest, the source sends the data along the interest gradient path. Another example is Rumor routing protocol, which uses a set of long-lived agents to create paths that are directed towards the events they encounter.

Negotiation based routing

Negotiation based protocols use high level data descriptors in order to eliminate redundant data transmissions through via negotiation. This idea aims to suppress duplicate information and prevent redundant data from being sent to the next sensor or the base-station. This is achieved by exchanging a series of negotiation messages before the real data transmission is carried out. One example of negotiation based routing is the family of SPIN protocols, which uses this mechanism and prevents from implosion and overlapping.

QoS-based routing

Networks with QoS-based routing protocols have to ensure some QoS factors, as for example, low delay, bandwidth, delivery, etc. when sending data to the base stations. However, this kind of protocols applied to WSN has to balance between energy consumption and data quality.

Some routing protocols bring several QoS features to WSN. For instance, in *Sequential Assignment Routing (SAR)* [35], routing decisions are made depending on three factors:

energy resources, QoS on each path, and the priority level of each packet. In order to provide energy efficiency and fault tolerance, SAR creates a tree from the source node to the destination nodes. However, this protocol suffers overhead with a high number of nodes. Another QoS routing protocol called *SPEED* [36] provides congestion avoidance and ensures a certain speed for each packet in the network, which ensures to estimate the end-to-end delay for the packets. However, SPEED does not consider any further energy metric in its routing protocol.

Coherent and non-coherent processing

WSN routing protocols employ different data processing techniques. Next are shown two different data processing techniques: coherent and non-coherent data processing-based routing. In non-coherent data processing routing nodes only process locally the raw data before sending it to other nodes, whereas in coherent data processing the minimum processing typically includes tasks like time stamping, duplicate suppression, etc. After this processing, data is forwarded to other nodes called aggregators for further processing.

Some examples of non-coherent and coherent processing are *Single Winner (SWE)* and *Multiple Winner (MWE)* algorithms, respectively. In the Single Winner algorithm, a single aggregation node with the highest energy reserves and computational capability is elected for complex processing. By the end of the SWE process, a minimum-hop spanning tree will completely cover the network. On the contrary, at the end of the MWE process, each sensor has a set of minimum-energy paths to each source node. MWE process obtains longer delay, higher overhead and lower scalability than non-coherent processing.

2.5.3.3 Comparison of features between protocols

Many of the described protocols fit under more than one category. The next table summarizes the main features of these WSN routing protocols:

Table 2. Classification and comparison of routing protocols in WSN [30]

	Classification	Mobility	Position Awarenes	Power Usage	Negotiation based	Data Aggregation	Localization	QoS	State Complexity	Scalability	Multipath	Query based
SPIN	Flat	Possible	No	Limited	Yes	Yes	No	No	Low	Limited	Yes	Yes
Directed Diffusion	Flat	Limited	No	Limited	Yes	Yes	Yes	No	Low	Limited	Yes	Yes
Rumor Routing	Flat	Very Limited	No	N/A	No	Yes	No	No	Low	Good	No	Yes
GBR	Flat	Limited	No	N/A	No	Yes	No	No	Low	Limited	No	Yes
MCFA	Flat	No	No	N/A	No	No	No	No	Low	Good	No	No
CADR	Flat	No	No	Limited	No	Yes	No	No	Low	Limited	No	No
COUGAR	Flat	No	No	Limited	No	Yes	No	No	Low	Limited	No	Yes
ACQUIRE	Flat	Limited	No	N/A	No	Yes	No	No	Low	Limited	No	Yes
EAR	Flat	Limited	No	N/A	No	No		No	Low	Limited	No	Yes
LEACH	Hierarchical	Fixed BS	No	Maximum	No	Yes	Yes	No	CHs	Good	No	No
TEEN & APTEEN	Hierarchical	Fixed BS	No	Maximum	No	Yes	Yes	No	CHs	Good	No	No
PEGASIS	Hierarchical	Fixed BS	No	Maximum	No	No	Yes	No	Low	Good	No	No
MECN & SMECN	Hierarchical	No	No	Maximum	No	No	No	No	Low	Low	No	No
SOP	Hierarchical	No	No	N/A	No	No	No	No	Low	Low	No	No
HPAR	Hierarchical	No	No	N/A	No	No	No	No	Low	Good	No	No
Sensor aggregate	Hierarchical	Limited	No	N/A	No	Yes	No	No	Low	Good	No	Possible
TTDD	Hierarchical	Yes	Yes	Limited	No	No	No	No	Moderate	Low	Possible	Possible
GAF	Location	Limited	No	Limited	No	No	No	No	Low	Good	No	No
GEAR	Location	Limited	No	Limited	No	No	No	No	Low	Limited	No	No
SPAN	Location	Limited	No	N/A	Yes	No	No	No	Low	Limited	No	No
MFR, GEDIR	Location	No	No	N/A	No	No	No	No	Low	Limited	No	No
SAR	QoS	No	No	N/A	Yes	Yes	No	Yes	Moderate	Limited	No	Yes
SPEED	QoS	No	No	N/A	No	No	No	Yes	moderate	Limited	No	Yes

2.6 Conclusions

The WSN architecture introduces several limitations to the network implementation. For that reason, the election and the development of suitable MAC and routing protocols for the future application scenario is the main objective. Furthermore, the correct integration of physical layer, MAC layer and routing layer will contribute to obtain better performance and behavior.

The diverse MAC and routing protocols described in this chapter provide different features and advantages. Consequently, it is very important to know the application scenario requirements and constraints in order to develop an adequate architecture and obtain appropriate results.

This election and implementation will depend significantly on our WSN scenario and application. The correct integration of physical layer, MAC layer and routing layer will contribute to obtain better performance and behavior.

The analysis about WSN protocols carried out during this chapter showed that the simplicity of CSMA, together with its well-know behavior and adequate performance, makes this protocol the right candidate for the simulation scenario of this project. Regarding the routing protocols, LEACH appears as a suitable protocol for the current research due to its clustered type organization, the balance of the overall energy consumption that it carries out and its relevance on the current researches.

3 WSN Simulators

3.1 Introduction

A **network simulator** is defined as a piece of software or hardware that predicts the behavior of a network, without a real network being present. This kind of tools help us to understand, predict the behavior and the results, find, correct and overcome mistakes in our network without the need of implementing the network and tuning it directly. Through the use of network simulators it is possible to obtain faster and better results without working with the real network, i.e., it helps us to save time, resources and costs.

3.2 Simulator requirements

As we mentioned on the previous chapter, WSN networks introduce new characteristics but also new constraints to our implementation. Hence, simulating WSN includes more specific properties to reflect the real behavior and obtain better results. Next are presented the requirements that network simulators should address. These requirements [37] are divided into *non-functional* and *functional requirements*.

- Non-functional requirements: these requirements provide ease of use, comfort and better interactivity to the users. Some of these requirements are:
  - *Open source*: this allows to the user develop their own modules.
  - *Platform independence*: this avoids the obligation of using a specific platform or Operating System.
  - *Visualization module*: a friendly user interface which provides graphical and dynamic information about the scenario and graphical results helps the user to understand the model and interact with the simulator.
- Functional requirements: these requirements provide more realism to our network model. Some of the functional requirements are:
  - *Hardware simulation*: it reflects the performance of sensor components like CPU, transceiver and sensor unit.
  - *Battery and Power models*: it shows the energy consumption and remaining energy levels.
  - *Propagation modeling*: a variety of propagation models like RF, optical communication and/or infrared results very appreciated.
  - *Protocols modeling*: the larger number of protocols developed the higher flexibility. It is also very useful and an effective approach to provide an API for defining new protocol in a simulator.
  - *Physical environment modeling*: the implementation of different propagation characteristics from diverse materials like soil, water or cement will provide more realistic results.
  - *Emulation*: the network emulation and the environment emulation provide better understanding of the network behavior and more accurate results.

3.3 A model for WSN simulation

This subsection describes a general component model for WSN simulation tools. The models include new components, not present in classical network simulators.

3.3.1 Network model

In the network model the next components are considered. Figure 13 shows a general network model.

- Nodes: each node is a physical device monitoring a set of physical variables. Nodes communicate with each other via a common radio channel.
- Environment: this component models the generation and propagation of events that are sensed by the nodes, and trigger sensor actions, i.e. communication among nodes in the network.
- Radio channel: it characterizes the propagation of radio signals among the nodes in the network.
- Sink nodes: these nodes interrogate sensors about an event of interest, receive data from the net, and process it.
- Agents: they are generators of events of interest for the nodes. The agent may cause a variation in a physical magnitude, which propagates through the environment and stimulates the sensor.

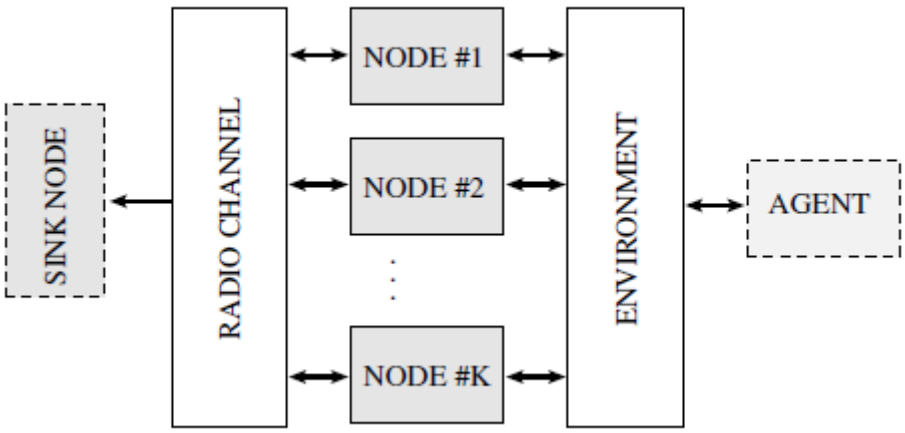

Figure 13. Wireless sensor network model

3.3.2 Node model

In order to provide a better description of the node behavior and their cross-layer interdependencies, the node model is divided into abstract tiers. Figure 14 depicts the node model.

- The *protocol-tier* contains the communication protocols. Two sub-components coexist at this tier: the protocol stack, which contains the MAC layer, and the routing layer, and a specific application layer component.

- The *physical-node* tier represents the hardware platform and its effects on the performance of the equipment. It is commonly composed of the set of physical sensors, the energy module and the mobility module.
- The *media-tier* connects a node with the environment through a radio channel and one or more physical channels.

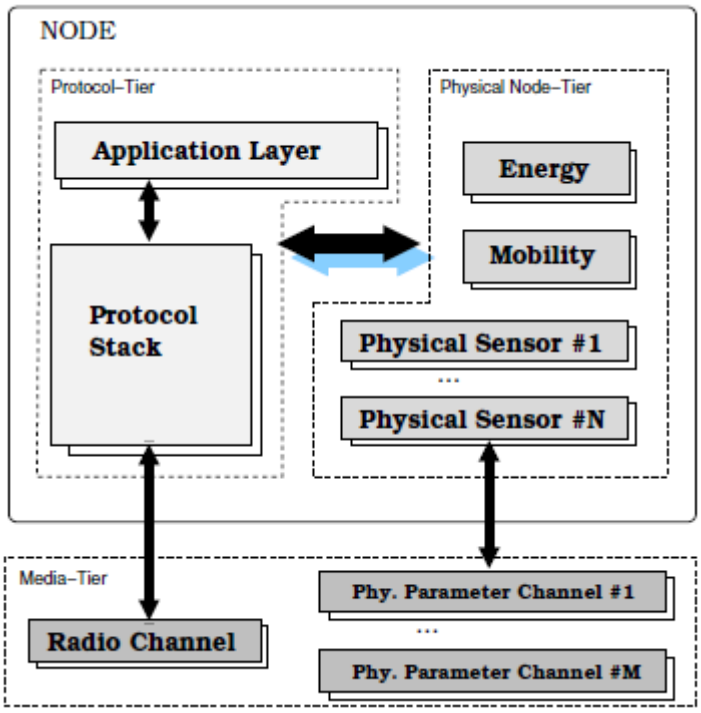

Figure 14. Tier-based node model

3.4 Network simulators

In this subsection, the most common network simulators are reviewed. These simulators provide different functionalities and capabilities.

3.4.1 The Network Simulator – ns-2

NS-2 [38] is a very popular general purpose discrete event simulation tool for sensor networks. Simulations are written in combination of C++ and OTCL (Object Tool Command Language), an object oriented scripting language. They can be observed graphically by Network AniMator (NAM). C++ is used for implementing protocols and extending the NS-2 library. OTCL is used to create and control the simulation environment itself, including the selection of output data. It supports WSN features like mobility model, wireless channel model and basic node energy model. These features can be improved and incremented by means of external applications or extensions (like MannaSim Framework, which introduces new modules for design, development and analysis of different WSN applications). The main drawback consists on the fact that NS-2 does not have good scalability for large sensor networks since exponential simulation time slowdown.

3.4.2 OMNeT++

OMNeT++ [39] is a public source component-based discrete event network simulator. It defines a simulation in a component-based architecture. Simulation models are described in C++ language and then assembled into larger components using NEtwork Description (NED) language to represent greater systems. The simulator has graphical tools for simulation building and evaluating results in real time. OMNeT++ scales well for very large scale network topologies, but without the proper simulation model or framework extensions, the simulator lacks suitable protocols and proper energy modeling for sensor networks. There are many extensions, frameworks and simulators for WSN based on OMNeT++ such as MiXiM, Castalia, Mobility Framework, EYES and etc. MiXiM provides detailed models of wireless channel (fading, etc.), wireless connectivity, mobility, obstacles and MAC protocols. Castalia is another extension with realistic MAC, wireless channel and radio model based on measured data. Mobility Framework extension implements the support for node mobility, dynamic connection management and a wireless channel mode. EYES is written for self organizing and collaborative energy-efficient sensor networks, which enables two-dimensional definition of the simulation map with different failing and error probabilities on different regions.

3.4.3 TOSSIM

TOSSIM [40] is a discrete event simulator designed and developed to simulate TinyOS wireless sensor networks. The TOSSIM architecture is composed of five parts. TOSSIM is also an emulator, as it can run the same simulated TinyOS application code on a real sensor. Furthermore, TOSSIM can simulate a mote's hardware, including digital I/O, ADC and sensors. Along with capability to simulate an application, operating system and network stack, TOSSIM is likely to provide more realistic results. With a detailed visualization module, results could then be easily understandable, but one drawback in TOSSIM is a lack of energy consumption modeling which is quite important in wireless sensor networks. There are few extensions for TOSSIM like TinyViz, a visualization tool, and PowerTOSSIM, an energy consumption modeling add-on.

3.4.4 OPNET

OPNET [41] is a commercial network simulator capable of simulating TinyOS applications. It enables scenario and statistics management which could not be found in TOSSIM. The models are the combination of OPNET specific code implementing TinyOS functionality and application specific code. This characteristic will reflect the interaction between the application and TinyOS. OPNET provides wide possibilities for wireless network simulations including WSN MAC protocols, very good accuracy on the radio transmission modeling and the possibility of modeling 3D outdoor scenarios. OPNET uses a hierarchical three level model to define each aspect of the system: the project editor, where network topology is designed; the node level, where individual network nodes and data flow models are defined; and the process editor, which uses a finite state machine approach to support specification of protocols, resources, applications and queuing policies. Finally, a simulation tool is included to support the three levels.

3.4.5 Ptolemy II

Ptolemy II [42] is an open-source software framework supporting experimentation with actor-oriented design, similar to component-based design. It includes Java packages that support different models of simulation paradigms (e.g. continuos time, dataflow, discrete-event). It also addresses the modeling, simulation and design of concurrent, real-time, embedded systems. VisualSense [43] is a modeling and simulation framework for WSN built on Ptolemy II. Models can be developed by subclassing base classes of the framework or by combining existing Ptolemy models. Thus, Ptolemy and its IDE components assure a simple and intuitive graphical composition of models.

3.5 Description of the OMNeT++ simulator

3.5.1 Overview

OMNeT++ with MiXiM plugin complies with the specific requirements and provides the required tools for the development of the project.

3.5.1.1 Modeling concepts

As we can look up on the OMNeT++ User Manual [44], OMNeT++ models consist of modules which communicate with message passing. The active modules are termed simple modules, and they are written in C++. Compound modules are made up of simple modules or other compound modules. Thus, the number of hierarchy levels is not limited. This architecture allows reusing the well-built modules and, moreover, allows implementing and customizing new modules with additional features. The whole model, called network, is also a compound module. The picture below shows the OMNeT++ network structure with different modules. Arrows connecting small boxes represent connections and gates.

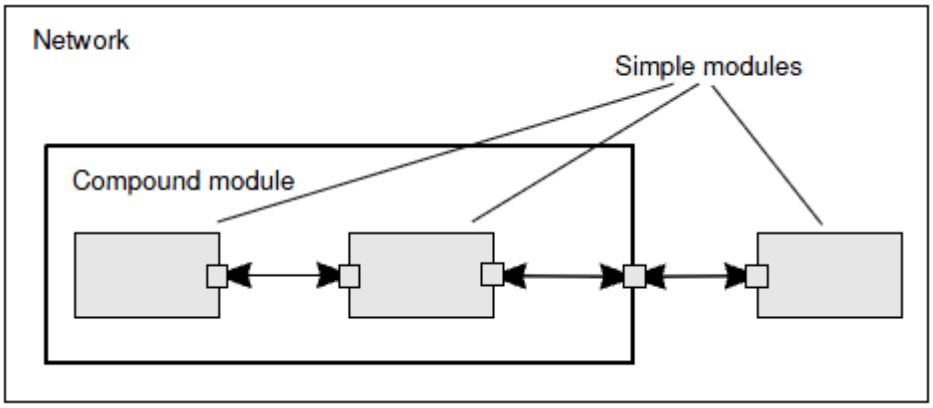

Figure 15. Simple and compound modules of an OMNeT++ network

Modules communicate with messages, which are typically sent via gates. Gates are the input and output interfaces of modules. An input and an output gate can be linked with a connection. Connections spanning across hierarchy levels are not permitted. Messages typically travel through a chain of connections, to start and arrive in simple modules. Parameters such as propagation delay, data rate and bit error rate, can be assigned to connections. Connection types with specific properties (termed channels) can also be defined and reused them in several places. Modules can have parameters. They can take

different values: string, numeric or boolean, and they are mainly used to pass configuration data to simple modules, and to help define model topology.

Thanks to its design and its structure, OMNeT++ provides efficient and useful tools for the user to describe the system. Some features of OMNeT++ are:

- hierarchically nested modules
- modules communicate with messages through channels
- flexible module parameters
- topology description language

3.5.1.2 The OMNeT++ simulation IDE

The OMNeT++ simulation IDE is an extension of Eclipse development platform with new editors, views and other functionality like tools for creating and configuring models and analyzing the simulation results. For further information, it's available an OMNeT++ IDE User Guide [45].

The Workbench

Eclipse is a very flexible system where you can manage different panels, editors and navigators.

The OMNeT++ IDE provides a "Simulation Perspective" to work with simulation related NED, INI and MSG files. Next are explained the meaning of these file extensions.

- The NED language topology description(s) (**.ned** files) describe the module structure with parameters, gates etc.
- The Message definitions (**.msg** files) define different message types and add data fields to them.
- The Configuration file (**.ini** files) contains settings that control how the simulation is executed, values for model parameters, etc.

The main window showed above on Figure 16 contains different and useful panels:

- The *Project Explorer* shows the projects and their content in your workspace.
- The *Properties View* contains the information on the object selected in the editor area
- The *Problems View* references the code lines where Eclipse encounters a problem
- The *Module Hierarchy*, *NED Parameters* and *NED Inheritance View* are with the network topology and its modules.
- The *Console View* shows the results of the executions.

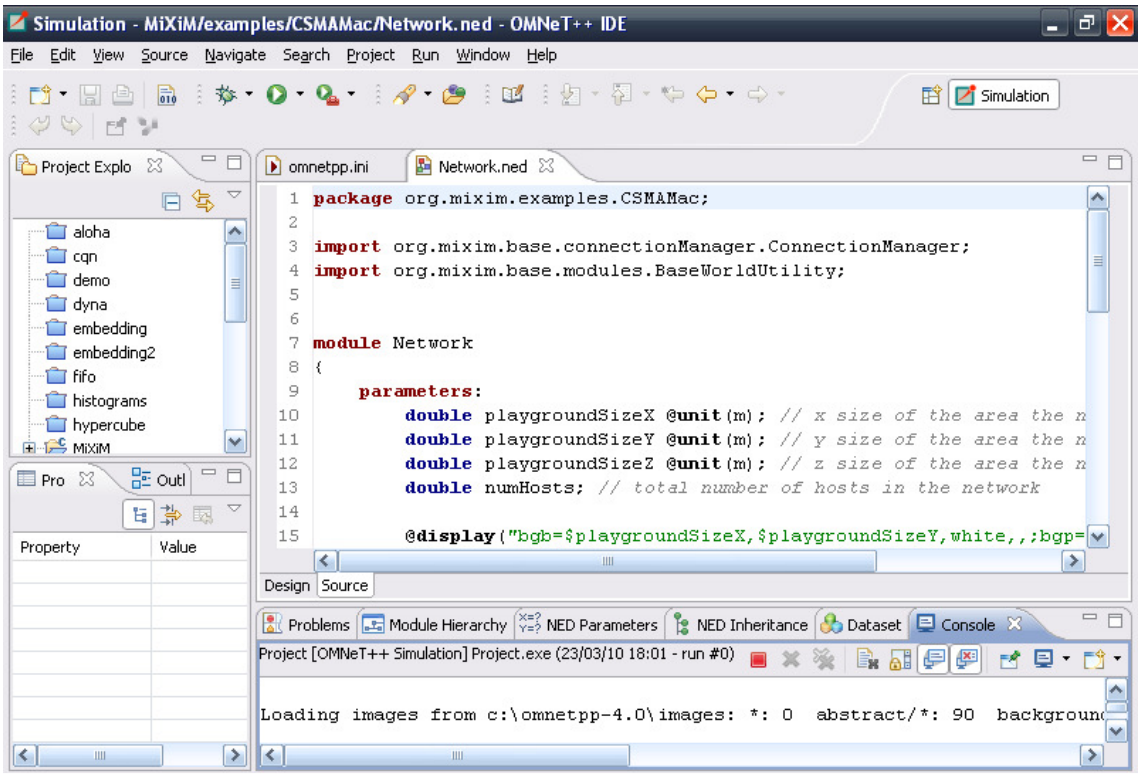

Figure 16. Default layout of the OMNeT++ IDE

The manner of creating models and networks is identical to the way of creating programs under Eclipse platform. The workspace is the directory where all your projects are located and can be linked.

Additionally, OMNeT++ provides specific editors for the simulations. The graphical NED editor, showed on Figure 17, helps to improve the visual network structure for a better understanding of the user, and INI file editor helps to edit the file with contains the configuration of simulation runs.

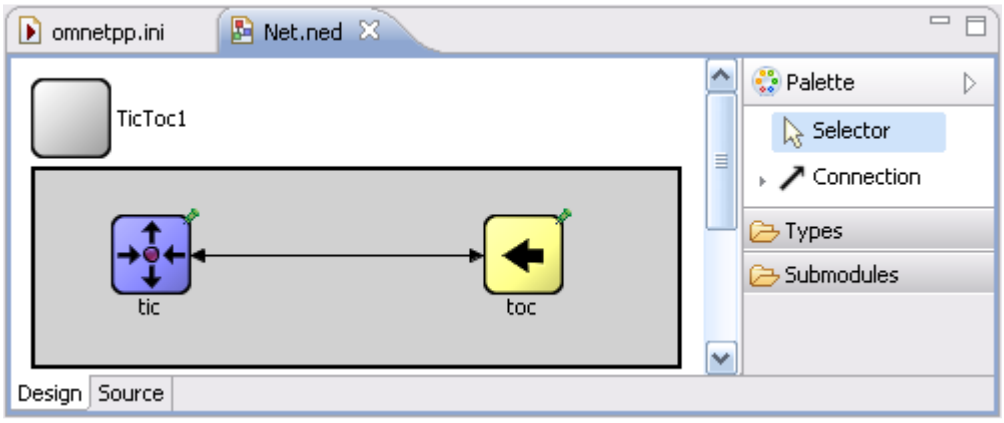

Figure 17. Graphical NED Editor

The Graphical Runtime Environment

The Tkend runtime environment, built in Tcl/Tk, is a portable graphical windowing user interface which supports interactive execution of the simulation, tracing and debugging. It is recommended in the development stage of a simulation since it allows one to get a detailed picture of the state of simulation at any point of execution and to follow what happens inside the network. Some important features are:

- message flow animation
- graphical display of statistics and output vectors during simulation execution
- event-by-event, normal and fast execution
- inspector windows to examine and alter objects and variables in the model
- scheduled messages can be watched in a window as simulation progresses

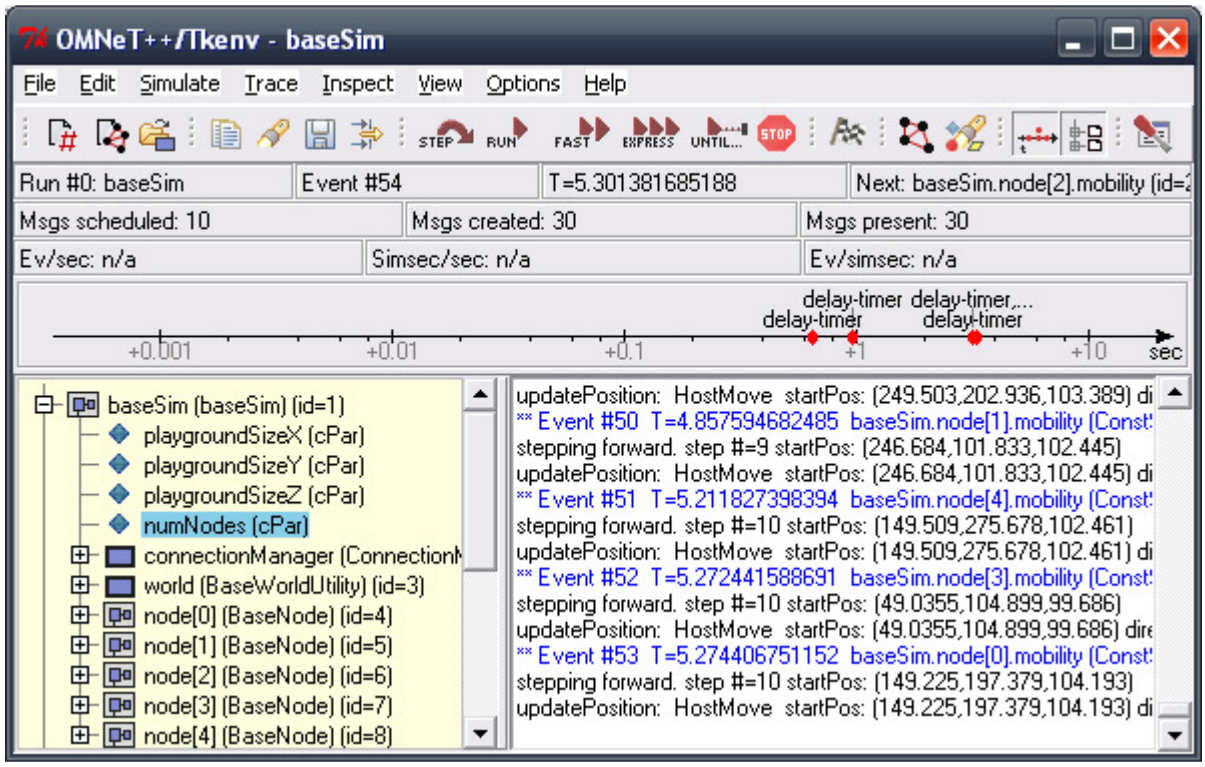

Figure 18. The main window of the Tkenv runtime environment

The main window of the Tkend environment showed on Figure 18 show different parts:

- The *toolbar* includes the access to the main functions of Tkenv like run, stop and start or finish the simulation and configure the visual appearance.
- The *status bar* contains the information about the current state of the simulation like the current run number and network name and the current event number, information about the number of messages and the events processed.
- The *timeline* displays the content of the FES on a logarithmic time scale.
- The *object tree* displays all inspectable objects currently present in the memory.
- The *log window* contains the output of the simulation. The window content can be filtered to include messages only from specific modules

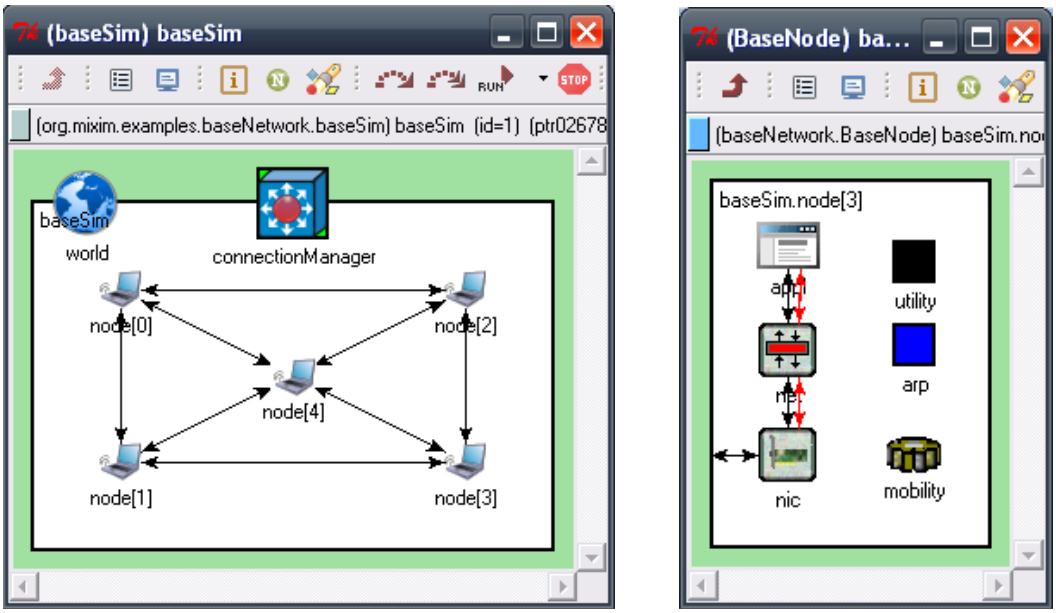

Figure 19. Top level network and node component structure

The top level network window shows the network structure. This window enables to explore the component hierarchy in a graphical mode. Networks and compound modules are represented by graphical inspectors displaying their internal structure. Each component can be inspected as object, as graphic or as module output and these different possibilities shows the component contents, its graphical internal structure and its output messages and events, respectively. The top level network and node component structure are showed above on Figure 19.

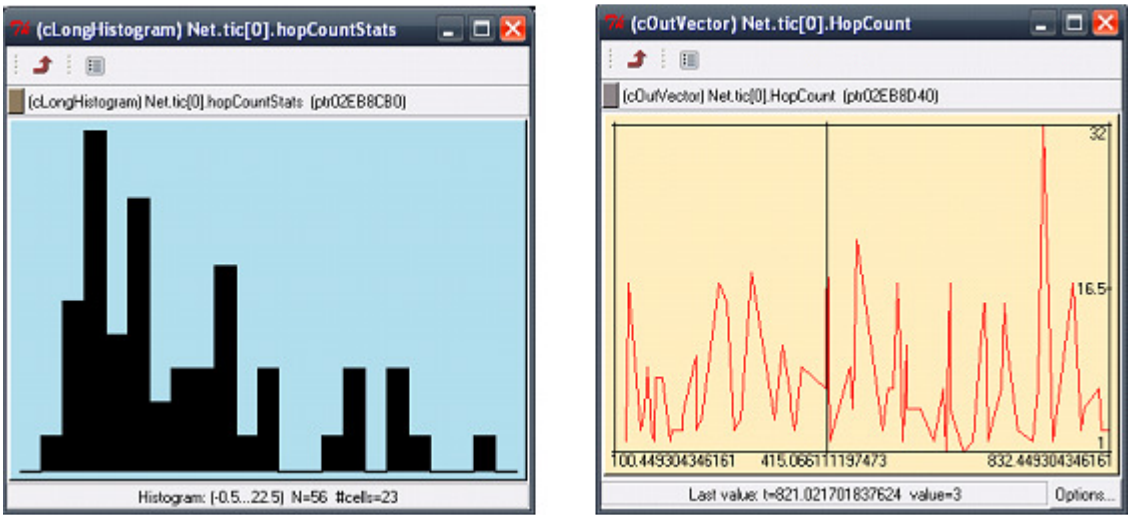

Figure 20. A histogram and an output vector

Furthermore, this environment can show output vectors in real-time as showed above on Figure 20. If any component contains an output vector, Tkenv will show a chart by double clicking on the object.

3.5.1.3 MiXiM

MiXiM [46] (mixed simulator) is a simulator for wireless and mobile networks using the OMNeT++ simulation engine. This simulator combines various simulation frameworks developed for wireless and mobile simulations in OMNeT++. MiXiM provides detailed models of the wireless channel (fading, etc.), wireless connectivity, mobility models, models for obstacles and many communication protocols especially at the Medium Access Control (MAC) level. Thus, MiXiM provides detailed models and protocols, as well as a supporting infrastructure which can be divided into five groups:

- Environment models: it reflects the relevant parts of the real world, such as obstacles or other elements which hinder wireless communication.
- Connectivity and mobility: the simulator tracks the movement of nodes and the variations on the influence between nodes and provides an adequate graphical representation.
- Reception and collision: the reception handling is responsible for modeling how a transmitted signal changes on its way to the receivers considering the movement of objects and nodes and transmissions making by other senders.
- Experiment support: it helps the researchers to compare the results and supports different evaluation methods.
- Protocol library: it enables researchers to compare and to share their ideas.

This simulator appears as the fusion of different simulator frameworks into one. These frameworks contribute with different approaches to MiXiM: the Mobility Framework (MF) with its mobility support, connection management, and general structure; the CHannel SIMulator (ChSim) with their radio propagation models; and the protocol library from the MAC simulator and the Positif framework.

Through its low memory consumption and its modular structure, MiXiM can support simulations with more than 1000 nodes. The graphical configuration interface helps to manage the model and it allows modifying some parameters, filtering the resulting information and adapting the level of detail and thus the execution time.

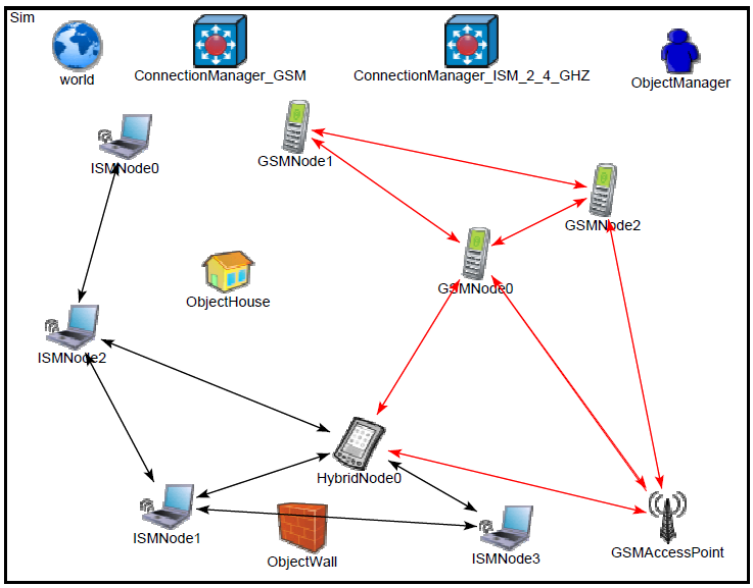

Figure 21. A network simulation

The general structure of MiXiM shows two different parts [47]:

- The simulation modules: a MiXiM network contains a "world" utility model which defines the environment properties like the size of the terrain, the kind of terrain simulation (2D or 3D) and different "objects" to model the environment of a simulation. The "ConnectionManager" module manages dinamically the connections between interfering nodes, where the signal quality is based on the interferences and the mobility. Finally, the "nodes" make up the network. MiXiM supports different kind of nodes (like Access Points and terminals) with different properties. An example of a MiXiM network is showed on Figure 21.
- The node structure: the nodes contain the modules according to the ISO/OSI architecture, together with other sensor specific and utile modules like the battery module, the mobility module, the arp module and the utility module, as it is showed on Figure 22. The layers of an IP model can be composed by the application layer (appl), the network layer (netw), the MAC layer (mac) and the physical later (phy). The physical and MAC layer are grouped into a Network Interface Card (NIC) module. The mobility module is responsible for the movements of a node or an object. The battery module is used to simulate the power consumption and properties. The arp module handles the Address Resolution Protocol (ARP), and the utility module provides a general interface for collecting statistical data of a simulation and maintains parameters that need to be accessed by more than one module within a node.

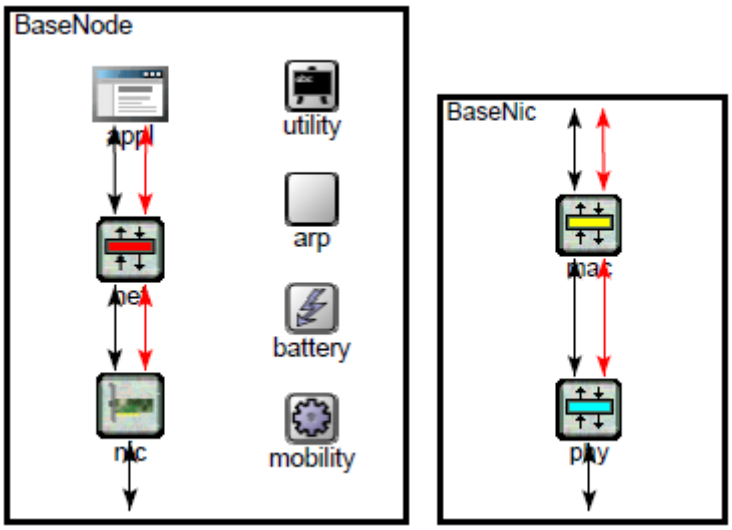

Figure 22. Node structure and NIC structure

3.5.2 Advantages

The combination of OMNeT++ with MiXiM provides numerous features which allow implementing a very detailed simulation in comparison with other WSN simulators.

Some of the best features of OMNeT++ with MiXiM are the differentiation between the network structure (implemented with NED) and the network behavior (implemented in C++). They also provide a set of important models like battery, power and propagation models, providing much more functionality and flexibility than other simulators with the lack of some of these modules like GloMoSim, SENS, ATEMU, Prowler and Shawn.

OMNeT++ and MiXiM are Open Source, have a very useful graphical support for debugging, support parallel simulation and show a very good scalability to large networks (more than 100 nodes), overtaking other simulators like NS-2 with worse scalability.

Furthermore, OMNeT++ is supported by a community site of software developers with several useful features like a mailing list.

3.5.3 Drawbacks

OMNeT++ with MiXiM show several drawbacks in comparison with other WSN simulators. The main drawback consists on the fact that OMNeT++ is a general purpose simulation framework and, for that reason, it only supports a limited emulation or Real-time OS/SW execution time modeling, unlike specific simulation tools like ATEMU, EmStar or TOSSIM. Other minor drawback consists on the lack of MAC protocols and the inexistence of any routing protocol. This drawback can be solved by the developer through the implementation of the required protocols.

OMNeT++ also doesn't provide a huge variety of MAC or routing protocols as other simulators like NS-2 or OPNET do. Because of this, the users need to resort to some extensions or some implementations done by the community.

3.6 Conclusions

After this chapter, we can ensure that OMNeT++ with MiXiM appear as a very good solution to implement and test the behavior of large scope wireless sensor networks. This simulation framework meets the majority of the simulator requirements explained above and also provides some important features for this project like very good scalability unlike other simulators do. The simulation structure facilitates the design and the test of large scope wireless sensor networks, its graphical runtime environment helps the user with the debugging and tracing tasks, providing better understanding of the network behavior, and MiXiM extension provides the advanced and specific modules required to simulate our sensor network.

4 Evaluation of routing protocols

4.1 Introduction

The routing protocol is a fundamental piece for the network operation. It will determine the network behavior. Furthermore, the network layer will add some additional features depending on the protocol implementation and the network requirements

During this section, the operation and implementation of two different protocols is offered. This fact will help to understand the diverse features that two different protocol implementations will provide. The first one, the Direct Transmission operation, is the simplest implementation of a routing protocol. The second one, the LEACH protocol has acquired great importance due to the fact that it has been the base for numerous network protocol improvements. In this manner, the protocol's behavior and its main parameters can be studied in order to obtain any improvement. In this chapter, the two mentioned algorithm are described and analyzed. This analysis will help us with a better understanding of the protocol's behavior for a further design and implementation into the chosen simulator, and therefore, to obtain a completely coherent protocol implementation from with the given description

4.2 Direct Transmission

4.2.1 Direct Transmission operation description

The Direct Transmission protocol is the simplest routing protocol. In the Direct Transmission protocol, the base station serves as the destination node to all the other nodes in the network as showed on Figure 23, where the end user can access the sensed data. The nodes only remain active during the data transmission to the base station. Consequently, won't spend energy on receiving the messages from the other nodes, but they will only spend the minimum data on listening the channel and, therefore, they will spend their battery capacity on sending messages to the base station.

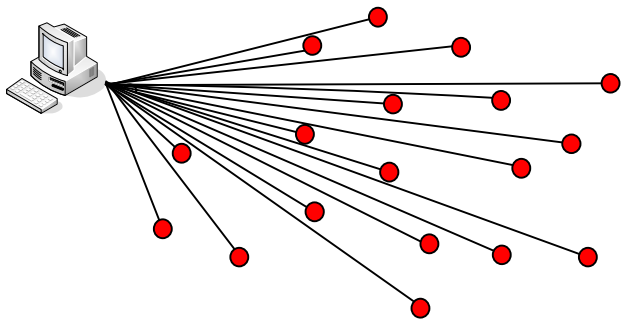

Figure 23: Network interconnection in Direct Transmission protocol

On the other hand, when a sensor node transmits data directly to the base station, the energy loss incurred can be quite extensive depending on the location of the sensor nodes relative to the base station. As a result, the Direct Transmission protocol's complexity can be negligible and its implementation quasi-trivial, but it is also the least energy efficient protocol in most cases.

4.2.2 Direct Transmission protocol implementation

4.2.2.1 Direct Transmission NED implementation

The Direct Transmission NED description (DirectTransmission.ned) is implemented as a *simple module* extending the **BaseLayer** module and implementing the **IBaseNetwLayer** interface[1].

```
simple DirectTransmission extends BaseLayer like IBaseNetwLayer
{
    parameters:
        @class(DirectTransmission);

        //Required *IBaseNetwLayer* parameters
        bool debug; // debug switch
        bool stats; // stats switch
        double headerLength @unit(bit);   // length of the network
                                          // packet header (in bits)
}
```

The first parameter that contains the Direct Transmission description is a direct reference to the Direct Transmission C++ class. The description also contains three parameters required by the IBaseNetwLayer interface and used by some Direct Transmission superclasses (from Direct Transmission C++ implementation).

4.2.2.2 Direct Transmission C++ implementation

The Direct Transmission C++ implementation consists of two different files: the **DirectTransmission.h** file, which contains forward declarations of variables, structures and subroutines, and **DirectTransmission.cc**, which contains the implementation the Direct Transmission protocol operation.

DirectTransmission.h file

The Direct Transission class implementation extends from the **BaseNetwLayer** class. The implementation of this simple protocol only contains one variable declaration:

```
int droppedMsgs;
```

This variable, also declared into the LEACH implementation, is a counter which accumulates the number of dropped messages that comes from the MAC layer. On the next chapter is detailed the whole utility and use of this variable.

Next are declared the subroutines required by Direct Transmission. These are divided in two parts: a public subsection and a protected subsection. The public subsection only contains one declaration:

```
virtual void finish();
```

This function performs value recording in some prefixed output files.

Finally, the protected subsection is declared with three subroutine declarations:

[1] For further information, visit the MiXiM API reference: http://mixim.sourceforge.net/doc/doxy/main.html

```
...
protected:
    /** @brief Handle control messages from lower layer */
    virtual void handleLowerControl(cMessage* msg);

    /** @brief Encapsulate higher layer packet into an NetwPkt*/
    virtual NetwPkt* encapsMsg(cPacket*);

    void handleHostState(const HostState& state) {} //does nothing
};
```

The first declaration, **handleLowerControl**, handles the message that comes from the lower layer, i.e., the MAC layer. The second declaration, **encapsMsg**, is in charge of encapsulating a higher packet layer into a network packet with the associated header. And finally, a third subroutine is declared. **handleHostState** is in charge of handling the host state change when a change announce is received. As it is shown above, the **handleHostState** subroutine is fully defined and does nothing. This fact means that the network layer won't be affected by any host state change, but the declaration of this function is required.

DirectTransmission.cc file

The DirectTransmission.cc file implements the functionality of Direct Transmission protocol. This protocol, owing to the fact that implements only one-hop message transmission, it doesn't add any special routing feature and this protocol is quasi implemented by the **BaseNetwLayer** class. Therefore, it is only needed to add some modifications to obtain the implementation within the simulated WSN.

The unique class needed to complete the Direct Transmission operation is:

```
NetwPkt* encapsMsg(cPacket*);
```

This subroutine encapsulates the packet received from the upper layer into a network packet ready to send to the lower layer. Because of the fact that the subroutine implemented into the **BaseNetwLayer** class doesn't know how to discover[2] the base station network and MAC address, the implemented subroutine execute the packet encapsulation with the knowledge of the base station network and MAC address defined into the class **ExtendedAddress.h.**

Finally, the **handleLowerControl** and **finish** functions are described. The first one, **handleLowerControl**, handles the MAC control messages, but it develops a special management with the packets that indicates a packet dropped from the MAC layer: if handled control message kind is `BaseMacLayer::PACKET_DROPPED`, the `droppedMsgs` counter value will be increased by 1 in order to count the quantity of dropped MAC messages and measure the correct integration between the MAC and the network layer. The second function, **finish**, completes the monitor of the `droppedMsgs` variable by recording the final value into an output file at the end of the simulation.

[2] For further information, visit the BaseArp class at the MiXiM API reference: http://mixim.sourceforge.net/doc/doxy/main.html

4.3 The LEACH protocol

4.3.1 LEACH algorithm's description

LEACH appears as one of the first cluster-based protocols which achieve to distribute the energy load among the entire sensor network. The main feature of LEACH is based, contrary to static clustering, on the randomized rotation of local cluster base stations (also called cluster heads) in order to distribute the data gathering and high power transmission (to the base station) energy consumptions. In this manner, LEACH enables scalability and robustness for dynamic networks, and incorporates data fusion into the data gathering process to reduce the amount of data to be transmitted.

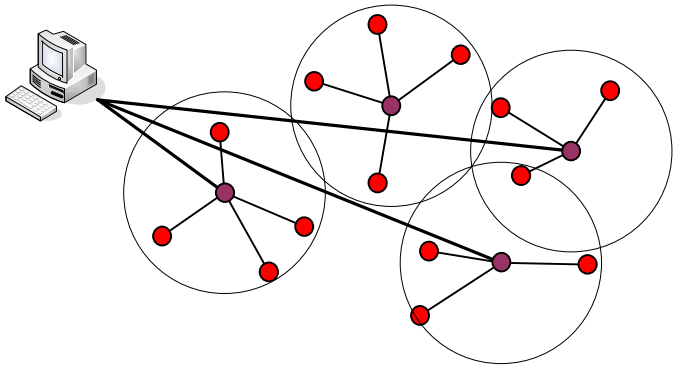

Figure 24: LEACH cluster type organization

Detailed below is a summary of LEACH algorithm description. The whole description of LEACH operation and further details can be found in [24] and [48]. The operation of LEACH is divided into *rounds*, and the rounds are also divided in different phases. Each LEACH round begins with a set-up phase, where cluster heads are randomly chosen and the cluster are organized as showed on Figure 24, and continues with a d steady-state phase, where nodes transmit their data to their respective cluster heads, and after that the cluster heads transmit the whole cluster "compressed" data to the base station. The phases that make up the algorithm operation are: Advertisement Phase, Cluster Set-Up Phase, Schedule Creation Phase and Data Transmission Phase (steady-state phase).

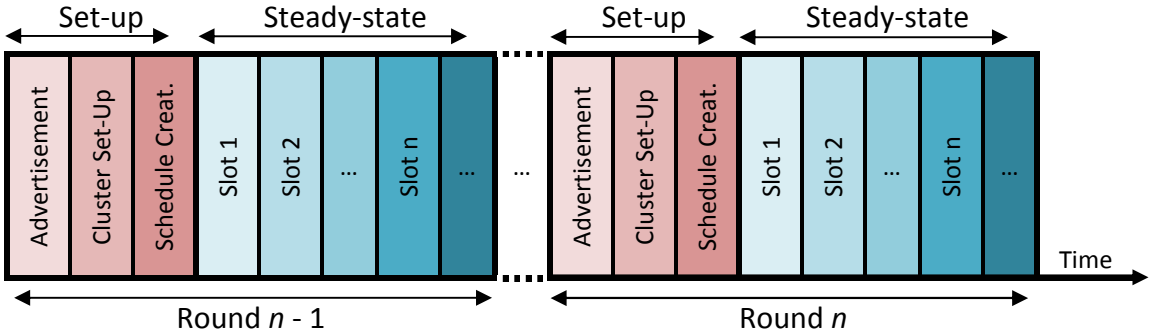

Figure 25. Time line showing LEACH operation

4.3.1.1 Advertisement Phase

When a new round begins, each node decides whether or not to become cluster head for the current round. This decision is made by the node *n* by choosing a random number between 0 and 1. The node becomes a cluster head if the randomly obtained value is less than a threshold *T(n)*. The threshold is set by the next formula:

$$T(n) = \begin{cases} \dfrac{P}{1 - P * \left(r \bmod \frac{1}{P}\right)} & \text{if } n \in G \\ 0 & \text{otherwise} \end{cases}$$

In the above formula, P = the desired percentage of cluster heads, determined a priori (e.g., 0.05), r = the current round, and G is the set of nodes that have not been cluster head in the last 1/P rounds. Thus, using this formula, each node will be cluster head at some point within 1/P rounds.

The nodes that have elected itself a cluster-head for the current round broadcasts and advertisement message to the rest of the nodes. This message is sent by using a CSMA MAC protocol and the same transmit energy (low power energy) for all the cluster-head nodes. The non-cluster-head nodes must keep their receivers on during this phase to hear the advertisements of all the cluster-head nodes.

4.3.1.2 Cluster Set-Up Phase

The non-cluster-head nodes decide the cluster-head to which it will belong for this round on the basis of the received signal strength of the advertisement message. The non-cluster head nodes will choose the cluster-head which sent the message with the largest signal strength heard. This fact means the election of the cluster-head to whom the minimum amount of transmitted energy is needed for communication.

After the decision is taken, each node must inform its respective cluster-head that it will be a member of the cluster. This message is sent to the cluster head by using a CSMA MAC protocol. For that reason, all cluster-head nodes must keep their receivers on.

4.3.1.3 Schedule Creation Phase

When each cluster-head has received all the messages for nodes that would like to be included in its cluster, they create a TDMA schedule. This schedule indicates when each cluster member can transmit. The schedule is broadcast back to the nodes in the cluster.

4.3.1.4 Data Transmission Phase

Once the TDMA schedule is fixed, nodes can transmit during their allocated transmission time to the cluster head if they have data to send. The radio of the nodes which are waiting to the node's allocated transmission time or the next round can be turned off in order to save energy. The cluster-head nodes must keep its receiver on to receive all the data from the nodes in the cluster. When all the data has been received, the cluster-head node performs data fusion tasks to compress the amount of data, and next this data is sent to the base station.

After a certain time (determined a priori), a new round begins with the Advertisement Phase as described in Section 0

4.3.2 LEACH algorithm's implementation

The development of LEACH under OMNeT++ requires a NED implementation (module description) and a C++ implementation (functionality description)

4.3.2.1 LEACH NED implementation

The LEACH NED description (LEACH.ned) is implemented as a *simple module* extending the **BaseLayer** module and implementing the **IBaseNetwLayer** interface[3].

```
simple LEACH extends BaseLayer like IBaseNetwLayer
{
    parameters:
        @class(LEACH);

        //Required *IBaseNetwLayer* parameters
        bool debug; // debug switch
        bool stats; // stats switch
        double headerLength @unit(bit);   // length of the network
                                          //packet header (in bits)

        //LEACH parameters
        double P;                    //percentage of CHs [0-1]
        double roundTime @unit(s);   //LEACH whole round time (in seconds)
        double slotTime @unit(s);    //slot transmission time (in seconds)
        double compressionIndex;     //Index of compression [0-1]
        double watingTime @unit(s); //Max time for going next stage (in s)
        int maxClusterSize;          //Maximum number of nodes per cluster

}
```

The first parameter that contains the LEACH description is a direct reference to the LEACH C++ class. The description also contains three parameters required by the IBaseNetwLayer interface and used by some LEACH superclasses (from LEACH C++ implementation).

Next appear the parameters which determine the LEACH protocol's behavior:

- **`P`**, the percentage of cluster-head nodes, is a real value between 0 and 1.
- **`roundTime`**, the LEACH entire round time, is a real value expressed in seconds.
- **`slotTime`**, the time for packet transmission within the Data Transmission Phase, is a real value expressed in seconds.
- **`compressionIndex`**, the index of data compression carried out by the cluster-head nodes, is a real value between 0 and 1.
- **`waitingTime`**, the maximum time that nodes wait to go to the next algorithm's phase, is a real value expressed in seconds.
- **`maxClusterSize`**, the maximum number of nodes that can contain a cluster, is an integer value.

[3] For further information, visit the MiXiM API reference: http://mixim.sourceforge.net/doc/doxy/main.html

4.3.2.2 LEACH C++ implementation

The LEACH C++ implementation consists of two different files: the **LEACH.h** file, which contains forward declarations of variables, structures and subroutines, and **LEACH.cc**, which contains the implementation of LEACH algorithm behavior.

LEACH.h file

The LEACH class implementation extends from the **BaseNetwLayer** class.

```
class LEACH: public BaseNetwLayer {
public:
      ...
      enum SelfMessages {     //Timers to go to different phases
            TIMER_NEW_ROUND, TIMER_JOIN_CH,
            TIMER_CREATE_TDMA_SCHEDULE, TIMER_SEND_DATA,
      };

      enum Phases {           //LEACH phases
            ADVERTISEMENT, CLUSTER_SETUP,
            SCHEDULE_CREATION, DATA_TRANSMISSION,
      };
      ...
```

The implementation begins with some `enum` structures. The first one showed above, `SelfMessages`, contains the different timers used during the operation. For instance, `TIMER_NEW_ROUND` defines the message kind when a new LEACH round starts after *roundTime* seconds, and `TIMER_SEND_DATA` defines the message kind when the slot to send data is reached. The second one, `Phases`, contains the different phases which passes the LEACH algorithm. This will help to avoid incoherences, for example, when a node receives a kind of message in a phase at which it shouldn't receive.

Next are declared the required variables for the correct implementation. The first three variables are `headerLength`, the length of the network packet header, `arp`, a pointer to the address resolution module, and `myNetwAddr`, the node network address. After these variables, common to any network layer implementation, the LEACH variables section begins. This section contains the needed variables to implement the LEACH algorithm.

```
    ...
    //-------------------------------------
    //      LEACH variables
    //-------------------------------------

    /** @brief Percentage of Cluster Heads */
    double P;

    /** @brief Round number */
    int currentRound;
    ...
    /** @brief Timer to go to the next stage */
    cMessage* timerNextStage;
    ...
```

Detailed below are the LEACH declared variables:

- **P**, **MAX_CLUSTER_SIZE**, **compressionIndex**, **roundTime**, **waitingTime** and **slotTime** are the variables to get the input parameters detailed in Section 4.3.2.1.
- **currentRound** indicates the current round number (the first round is 0), and **lastRoundCH** indicates the last round number when the node was cluster-head (-1 indicates never).
- **currentPhase** indicates the index of the current phase (referred to the `Phases` *enum* structure).
- **myCH** stores the address of the current cluster-head of a node. If the node is cluster-head during the current round, **myCH** will contain the BS address.
- **packetToSend** is a pointer to the next packet to send to the base station, i.e., the last packet received from the application layer.
- **distanceToCHs** is a map structure used by non-cluster-head nodes. It associates cluster-head node network addresses with the distance to them. This structure is required to choose the best cluster-head node, i.e., the nearest cluster-head node.
- **membersCH** and **netwQueue** are two lists used by cluster-head nodes. The first variable contains the addresses of all the cluster members. The second variable stores the packets of all cluster members. After the reception of all the packets, the data is compressed and sent to the base station.
- **timerNextRound** and **timerNextStage** are two pointers to message variables which work as timers. The first variable is employed to activate each new LEACH round. The second variable is employed to move a node within different stages when message gathering is required, e.g., when a non-cluster-head node collects all the cluster-head announces or a cluster-head collects all the "join" messages from non-cluster-head nodes.
- **droppedMsgs** is a counter which accumulates the number of dropped messages that comes from the MAC layer.

Finally, the subroutines are declared. This section is divided into three parts: a public, a protected, and a private subsection. The public subsection contains only two functions: **initialize** function, which is initializes the LEACH variables and their associated superclasses, and **finish** function, which performs output value recording tasks.

The protected subsection contains the functions related with message handling.

```
...
protected:

    /** @brief Handle self messages */
    void handleSelfMsg(cMessage* msg);

    /** @brief Handle messages from upper layer */
    virtual void handleUpperMsg(cMessage* msg);
    ...
    /** @brief Handle control messages from lower layer */
    virtual void handleLowerControl(cMessage* msg);
    ...
```

Next are described the declared functions:

- **handleSelfMsg** handles the messages sent to itself. The self message within the LEACH protocols work as timers, e.g., the timer used on launching a new LEACH round, or the timer used when the transmission slot is reached.
- **handleUpperMsg** and **handleLowerMsg** handle the messages which comes from the upper and the lower layer, i.e., from the application and the MAC layer.
- **encapsMsg** is in charge of encapsulating a higher packet layer into a network packet with the associated header.
- **handleLowerControl** handles the control messages that comes from the lower layer, i.e., the MAC layer.

The last subsection is the private subsection. It contains the function declarations associated to the LEACH operation.

```
...
private:

      /** @brief Calculates the threshold value to be Cluster Head */
      double calculateThreshold();

      /** @brief Executes the Advertisement Phase steps */
      void advertisementPhase();
      ...
      /** @brief Sets the MAC transmission power to low range*/
      void setLowTxPower();
};
```

Below are described the mentioned functions:

- **calculateThreshold** calculates the threshold for a node within the current stage. It is calculated on the beginning of the Advertisement Phase.
- **advertisementPhase**, **clusterSetUpPhase**, **scheduleCreation** and **dataTransmission** contains the protocol operation during the different phases. **advertisementPhase** and **dataTransmission** contains functionality for cluster-head and non-cluster-head nodes but **clusterSetUpPhase** only contains functionality for non-cluster-head nodes, and **scheduleCreation** functionality for cluster-head nodes. This is due to the fact that, for example, during the luster Set-Up Phase, only the non-cluster-head nodes initiate the phase and prepare a "join" message to send to the best cluster-head. Therefore, cluster-head nodes only must wait for the "join" messages. That's why their phase functionality is described into the **handleLowerMsg** function, when the message is received. In the case of the Schedule Creation Phase, the situation is the same but on the other way round with non-cluster-head and cluster-head nodes.
- **setHighTxPower** sets the transmitting power to a high value in order to make a long-distance cluster-head – base station transmission. **setLowTxPower** gets back the normal transmitting power.

LEACH.cc file

The LEACH.cc file implements the whole functionality of LEACH protocol.

The implementation of LEACH protocol is designed as follows: the LEACH network messages interchange carries out the cluster set-up, whereas the use of timers performs the switching between different phases or LEACH rounds. For a better understanding, the LEACH functionality has been divided into C++ functions with the same LEACH phase name. Therefore, the LEACH functionality is implemented by means of four functions that implement the majority of the protocol behavior, four functions that carry out different message handling tasks, and five functions of small functionality.

Before the protocol operation, the LEACH class requires the call of the next function:

```
void LEACH::initialize(int stage);
```

This function is called two times (during two initializing stages). During the first initialization stage, all the parameters from the NED file are got and the addresses, counters and other variables are initialized. During the second initialization stage, the timer for the first LEACH round is activated.

Detailed below are the functions which implement the LEACH functionality:

- **void LEACH::advertisementPhase():** this function implements the LEACH Advertisement Phase. The function starts with a call to the function **calculateThreshold**. If a random obtained value between 0 and 1 is less than the returned threshold, the node became cluster-head: it will update the `lastRoundCH` variable and will broadcast a "CH announce" message (with *CH_STATUS_BROADCAST* message kind, the node source address –`myNetwAddr`– and *L3BROADCAST* address as destination address). If the random value is higher, the node will be non-cluster-head: it will wait for "CH announces". Here a timer is activated (`timerNextStage`). If after a `waitingTime` the node don't receive any "CH announce", it goes to the next phase.
- **void LEACH::clusterSetUpPhase():** this function implements the LEACH Cluster Set-Up Phase. Within this phase, only the non-cluster-head node's behavior is implemented. Thanks to the `distanceToCHs` structure, the node chooses the nearest cluster-head node, sets `myCH` variable and sends it a "join CH" message (with *JOIN_CH* message kind, the node source address - `myNetwAddr`– and `myCH` address as destination address). After sending the message, non-cluster-head nodes wait to the "TDMA schedule" message. The Cluster Set-Up Phase functionality for cluster-head nodes is implemented into the **handleLowerMsg** function.
- **void LEACH::scheduleCreation():** this function implements the LEACH Schedule Creation Phase. Within this phase, only the cluster-head node's behavior is implemented, and they will create a "TDMA schedule" message (with *TDMA_SCHEDULE* message kind, the node source address –`myNetwAddr`– and *L3BROADCAST* address as destination address). This packet will contain a list with the cluster members' network addresses, i.e., the transmitting slot position associated to each cluster member. After broadcasting this packet, cluster-head nodes will wait to receive all the data packets from the cluster members to perform data compression and send the data to the BS. The Schedule Creation

Phase functionality for non-cluster-head nodes is implemented into the **handleLowerMsg** function.

- **void LEACH::dataTransmission():** this function implements the LEACH Data Transmission Phase. For the non-cluster-head nodes, when the transmission slot is reached, the procedure consists on setting the packet addressee and sending the packetToSend stored packet if a new application packet has been received, and waiting for a new LEACH round. When cluster-head nodes start to execute this function, they will have received all the non-cluster-head node data packets. Therefore, they get all the data from the stored packets and apply data compression. This process is simulated by creating a data packet whose packet size will be the application of the compression index after the sum of all the cluster member stored data packet sizes. This packet will have the same message kind of any of the data stored packets, the cluster-head node source address –myNetwAddr– and *L3BS* address as destination address. Before sending the packet, the cluster-head nodes call the **setHighTxPower** function to be able to reach the BS. After sending the packet, cluster-head nodes switch to normal transmitting power and wait for a new LEACH round.

Next are described the message handling functions:

- **void LEACH::handleUpperMsg**(cMessage* msg)**:** this function only calls the **encapsMsg** function with the received application packet as parameter, and stores the network packet resulting from the call. The message will be sent on the next transmission slot if the application layer doesn't send another message before. In this case, the last stored message will be overwritten.
- **void LEACH::handleLowerMsg**(cMessage* msg)**:** this function handles the packets that comes from the MAC layer. As it was mentioned above, the functionality of some phases is implemented into this function. Therefore, this function is responsible of the correct communication between cluster-head and non-cluster-head nodes. The function's behavior depends on the message kind:
  - *CH_STATUS_BROADCAST* messages should be managed by non-cluster-head nodes during the Advertisement Phase. They, by means of a "special" class, **Distance.h**, calculate the distance to the message source, which is a cluster-head node, and stores it into the distanceToCHs structure. After that, they reactivate the timerNextStage with a new waitingTime to receive new "CH announces" or going to the Cluster Set-Up Phase. If a cluster-head node receives this message, it just deletes it.
  - *JOIN_CH* messages should be managed by cluster-head nodes during the Cluster Set-Up Phase. They save the message source into the membersCH address if the MAX_CLUSTER_SIZE is not reached, which means that this node will be a cluster member. After that, they reactivate the timerNextStage with a new waitingTime to receive new "Join CH" messages or going to the Schedule Creation Phase. If a non-cluster-head node receives this message, it just deletes it.
  - *TDMA_SCHEDULE* messages should be managed by non-cluster-head nodes during the Schedule Creation Phase. If the message source is it cluster-head node, the non-cluster-head node look for its transmission slot. If it

doesn't find the slot, it means that the node doesn't belong to any cluster and it must wait for the next LEACH round. If the node finds its slot, it should switch its state to *SLEEP* mode and wait for its transmitting slot (activates the `timerNextStage` with `my_slot*slotTime` time). If the node has the first slot, it just calls the **`dataTransmission`** function. If a cluster-head node receives this message, it just deletes it.

- The rest of the messages are supposed to be *DATA_MESSAGE*. Therefore, only cluster-head nodes will handle this kind of messages during the Data Transmission Phase, and they will just store the message into the `netwQueue` structure. If a non-cluster-head node receives this message, it just deletes it.

- **`void LEACH::handleSelfMsg`**`(cMessage* msg)`: this function is in charge of handling self messages, which are used as timers in the OMNeT++ model. Timers are activated by cluster-head and non-cluster-head nodes depending on the phase at which they are:
  - When *TIMER_NEW_ROUND* is received, which is activated by all the nodes, a new LEACH round starts. `currentRound` counter is increased by 1, host state is switched to *ACTIVE*, `timerNextRound` is activated again with `roundTime` time and the **`AdvertisementPhase`** function is called.
  - When *TIMER_JOIN_CH* is received, means that non-cluster-head nodes won't receive any more "CH announce" message. If nodes have received any "CH announce", they switch to the Cluster Set-Up Phase.
  - When *TIMER_CREATE_TDMA_SCHEDULE* is received, it means that non-cluster-head nodes won't receive any more "Join CH" message. Therefore, they switch to the Schedule Creation Phase with a **`scheduleCreation`** function call.
  - When *TIMER_SEND_DATA* is received, which is activated by both cluster-head and non-cluster head nodes, means that their transmission turn has arrived. Host state is switched to *ACTIVE* mode, the transmission begins with the **`dataTransmission`** function call, and host state is switched again to *SLEEP* mode until the next round arrives.

- **`void LEACH::handleLowerControl`**`(cMessage* msg)`: this function handles the MAC control messages. If the handled control message kind is `BaseMacLayer::`*PACKET_DROPPED*, `droppedMsgs` counter value will be increased by 1.

For the rest of the functions, the description given into the above section is enough.

Finally, when the simulation has finished, a call to the next function is made:

`void LEACH::finish``():`

This function only records the `droppedMsgs` value into an output file.

4.4 Conclusions

After choosing the protocols that are interesting to research about any improvement, it is important to study how they work and how they behave. The analysis of each protocol phase provides a global idea about how it works but also a near knowledge about the operation step by step.

As it has been shown, the design and implementation of a new protocol is not a trivial task. A correct design will provide a good codification architecture, which also will help with a better understanding and ease of debugging. The implementation model should be clear and it should use the minimum required resources in order to obtain efficient results when the protocol operation is moved to a real implementation.

The two different protocols presented, Direct Transmission and LEACH, show a completely different behavior. Whereas Direct Transmission offers simplicity, LEACH provides advanced features as scalability, robustness and energy savings in most cases at the expense of a more complex protocol implementation and network organization. These features of each protocol are reflected into the simulation protocol's implementation: the Direct Transmission protocols requires a simply implementation, whereas LEACH implementation requires a more complex node organization, message handling and data management.

5 Simulation scenarios

5.1 Introduction

When a research requires performing any simulation, a description of the simulation scenario and specific requirements is needed. A description of the simulation scenario will help to understand the problem and address the solution in a more efficient way. Furthermore, a detailed description allows reproducing the simulation scenario in different simulation environments for any improvement or comparison study.

The implementation of the simulation scenario of this research will be build over the MiXiM framework. This fact will help to reuse the framework modules, will provide a faster and more robust implementation and, extending the framework functionality, the network behavior and simulation results could be adapted to our requirements.

5.2 Description of the simulation scenarios

Direct Transmission is a protocol for WSN which offers simplicity and acceptable results in specific circumstances and scenarios, and LEACH protocol is a self-organizing protocol which provides good scalability. For this reason, the aim of the present research consists on the study of the behavior of large scope WSNs with different routing possibilities.

The present document will study the behavior of a network where nodes send data to the base station with different network protocols. In general terms, the network will be made up of a hundred nodes scattered randomly in a 1000×1000m area and a fixed base station. The network nodes will contain a battery module with a predetermined battery power and they will also have some constant speed mobility. The node radio transmitting power will be 500mW, but the data transmissions to the base stations will require 10W power. The MAC layer will use CSMA MAC protocol and the application layer will generate data in a regular manner with a predetermined data generation period and a prefixed data size. A further simulation scenario description can be found in Appendix I. Table of Specifications of the simulation scenarios.

The simulations will obtain diverse results from two different routing protocols: Direct Transmission and LEACH. Furthermore, the behavior within both two protocols will be studied by means of changing the value of some important parameters like the data size, the time between two sensing acts or the speed of the nodes.

5.3 Design and Implementation of the simulation scenarios

The simulation scenarios have been built over the MiXiM framework. In this manner, a typical structure from a MiXiM example simulation has been taken.

5.3.1 Implementation of a network simulation step by step

The network simulation implemented in this document follows the implementation architecture of MiXiM framework. Therefore, the general steps for the creation of a MiXiM network are detailed[1]:

- For the creation of an OMNeT++ Project, follow the next indications from the menu bar: **File → New → OMNeT++ Project…**
  In the new window, select the desired project name and push "Finish" button.

- For the use of the MiXiM framework in the implementation, the MiXiM resources should be included following the next indications from the project's pop-up menu: **Properties → Project References → Select "MiXiM" project.**

- The MiXiM framework contains a basic network implementation. This implementation can serve as base for specific implementations. Therefore, the network files should be copied from the next MiXiM folder[2]:
  . /MiXiM/examples/baseNetwork

- For the creation of specific modules, the following files should be created:
  - Network Description File (.ned file): this file will contain the specific parameters, submodules, gates, etc. It can extend, and thus, specialize parent modules. For the file creation from the pop-up menu:
    New → Network Description File (ned)
  - C++ class: following the programming principles, two files should be created: a header class, with the variables and subroutines declaration, and a source file, with the codification of the behavior. For the file creation of these files from the pop-up menu: **New → Class (OMNeT++)**

- For the integration of the created modules into the network simulation, the network description file (**BaseNetwork.ned**) or the node description file (**BaseNode.ned**) must be edited with the specific module inclusion.

- The following consists on editing the content of the configuration file (**omnetpp.ini**) should be edited with the initialization of the network parameters and the declaration of the specific modules created for the network simulation.

- The last step consists on running the simulation of the network created. Before launching the execution, the project must be built following the next indications from the menu bar or the pop-up menu: **Project → Build Project**
  For the simulation execution, the next steps should be followed from the omnetpp.ini file's pop-up menu: **Run as → OMNeT++ Simulation…**
  The Tkenv runtime environment will appear with the created simulation.

[1] For information about the OMNeT++ operation basics and the implementation of simple networks visit: http://www.omnetpp.org/doc/omnetpp41/tictoc-tutorial/

[2] For further information about the base network MiXiM implementation visit: http://sourceforge.net/apps/trac/mixim/wiki/HowToStart

5.3.2 Implementation of the simulation network architecture

Our simulation scenarios have been built from the **baseNetwork** simulation example. The network structure is defined into the **BaseNetwork.ned** as showed below:

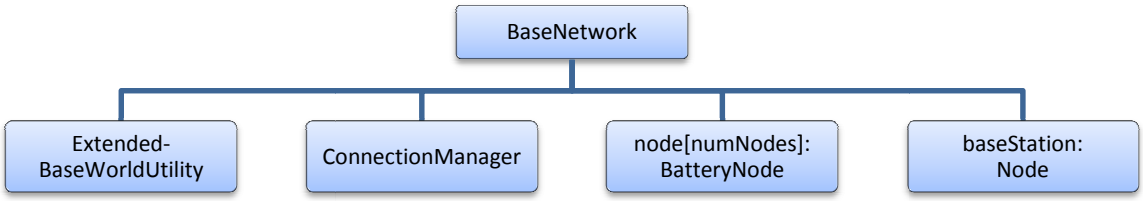

Figure 26. Network simulation structure

All the modules from the above architecture are from MiXiM framework. Next is described their function:

- **BaseNetwork:** is the simulation parent module (the *System* module).
- **ExtendedBaseWorldUtility:** contains global utility methods and parameters like playground size. Extends from **BaseWorldUtility** MiXiM's framework (adds some value recording at the end of the simulation, as the number of alive nodes during the network lifetime).
- **ConnectionManager:** checks if any two hosts can hear each other and updates their connections accordingly. If two hosts are connected and can hear anything, it doesn't mean that they can understand it.
- **BatteryNode:** defines the node's structure of our simulation.
- **Node:** defines the base station's structure.

The node structure is defined into the **BaseNode.ned** file (and the file **BaseNic.ned** for the NIC definition). The node network architecture is organized as follows:

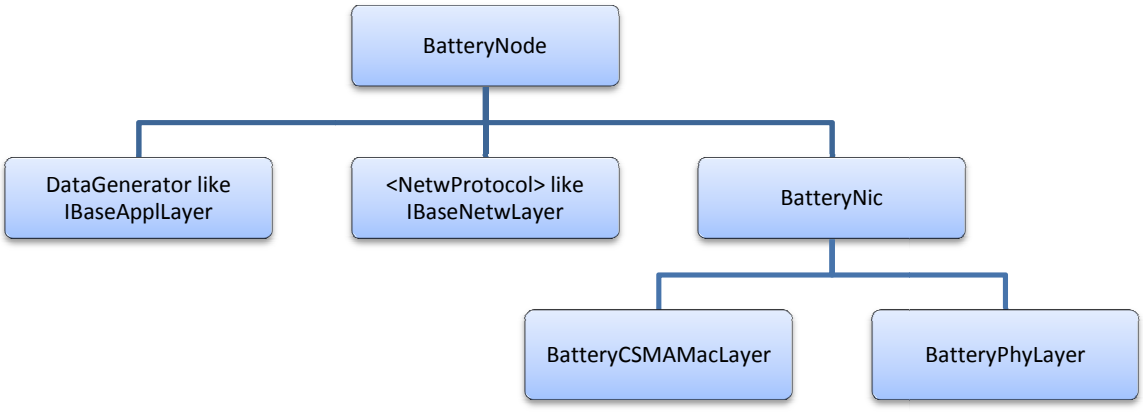

Figure 27. Node network structure

The functionality of the node network modules is detailed below:

- **DataGenerator:** is responsible of "sensing" and data generation. The sensing period (time between two sensing samples) and other values like the data size must be given as input parameters.

- **<NetwProtocol>:** carries out the networking functions. Into this module will be present two different possibilities in different simulations: **DirectTransmission** and **LEACH**. The complete functionality of these two possibilities within this module is detailed in Section 4.2 and 4.3 respectively.
- **BatteryNic:** it simulates the Network Interface Controller (NIC). It is made up of the MAC and the PHY layer:
  - **BatteryCSMAMacLayer:** implements the CSMA MAC protocol. Extends from **CSMAMacLayer** MiXiM's framework (adds some transmitting power switching capacity required by LEACH to make high power transmission to the base station).
  - **BatteryPhyLayer:** implements the physical layer. Extends from **PhyLayer** MiXiM's framework (adds some parameters for energy consumption calculation[3]).

The node network modules are interconnected similarly as the OSI network architecture. Therefore, when a message is sent, it will start from the application layer; will go through the network and MAC layer until it reaches the physical layer. When a message is received, the inverse procedure takes place.

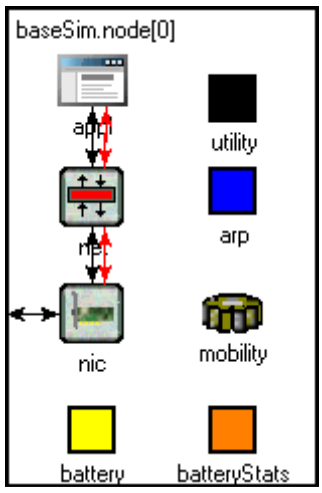

Figure 28. Internal node structure

Internally, the MiXiM framework implements this behavior by means of the *handle message* functions. For instance, when a layer sends a packet to the lower layer through the `sendDown` call, the lower layer will handle the message by means of the `handleUpperMsg` function, and when a layer sends a packet to the upper layer through the `sendUp` call, the upper layer will handle the message by means of the `handleLowerMsg` function. For the control message interchange, the operation is the same, but using the `sendControlDown`, `sendControlUp` calls and the `handleUpperMsg` and `handleUpperControl` functions.

[3] A special importance for the evaluation of the simulation scenario has the energy consumption model. For further information, see Section 6.2

Furthermore, the nodes contain some other functionality:

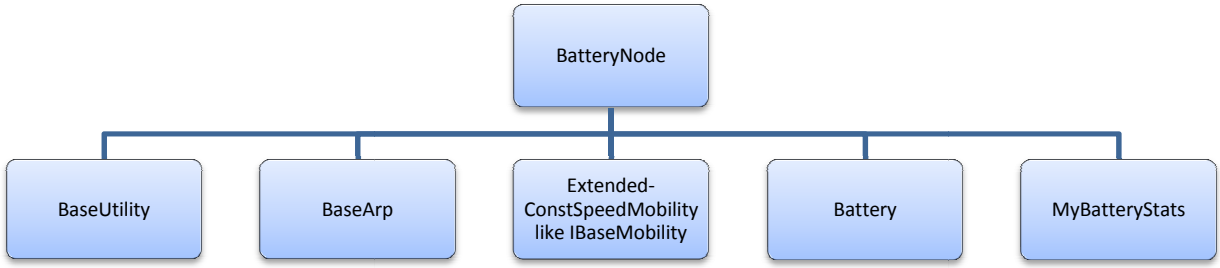

Figure 29. Mobility and utility node modules

Next is detailed the battery, mobility and utility node modules:

- **BaseUtility:** this mandatory module contains node wide utility methods, mainly a black board like subscribe and publish feature which is used to publish statistic information.
- **BaseArp:** used by **BaseNetwLayer** and **BaseMacLayer** for address resolution.
- **ExtendedConstSpeedMobility:** IBaseMobility is a mandatory module which defines current position and the movement pattern of the node. The selected class extends from **ConstSpeedMobility** MiXiM's framework, which defines a linear constant speed mobility (adds some minor required functionality).
- **Battery and MyBatteryStats:** implements a simple battery module and a utile module for collecting battery statistics. The battery capacity and other values must be given as input parameters

The base station structure is quite similar to the node structure. The base station network architecture is also made up of the application layer, the network layer, the MAC layer and the physical layer. The differences consist on the application and the network layer: the base station is only data receiver (data drain). Therefore, its application layer won't generate any data (this class will be called **BSApplLayer**), and its network layer won't route any packet: they will only handle messages coming from the lower layer (this class will be called **BSNetwLayer**). Furthermore, the base station doesn't have battery constraints. Thus, the base station will use the **BSCSMAMacLayer**, a simple subclass of **CSMAMacLayer** which recognizes the BS MAC and network addresses, and the **PhyLayer** module.

Owing to the base station doesn't have battery constraints; it doesn't implement a battery module and battery statistics collector module. Due to the mobility module is mandatory for the node implementation, the base station will use the **ExtendedConstSpeedMobility** implementation with a speed value of 0.

5.3.3 Initialization of the module network parameters

The initialization of the network parameters is made through the **omnet.ini** file. This is a simulation *initialization* file required to set the initial simulation values. Next is shown a brief example of the simulation's **omnet.ini** file:

```
[General]
cmdenv-config-name = perftest
cmdenv-express-mode = true
ned-path = ../../../MiXiM/base;../../../MiXiM/modules; {...}
network = baseSim

##########################################################
#                  Simulation parameters                 #
##########################################################
num-rngs = 1
seed-0-mt = 1200
baseSim.playgroundSizeX = 1000m
baseSim.playgroundSizeY = 1000m
baseSim.playgroundSizeZ = 100m
baseSim.numNodes = 10
...
```

The **omnet.ini** file is divided into different sections. The first and mandatory section is the `[General]` section. This section will contain the default values for all the parameters of the network modules. Furthermore, the *General* subsection is divided into the next subsections[4]:

- Simulation parameters: contains general simulation parameters for the parent (System) simulation module, e.g., number of random number seeds, simulation playground size, etc.
- WorldUtility and Channel parameters: contains parameters that detail the node interconnection within the simulation scenario, like the carrier frequency of the channel or the signal attenuation threshold.
- Base station and node common module parameters: contains the common values for the parameters that base station and nodes share. For instance, the physical and MAC layer specific values for a correct communication, or the MAC, network and application header size to make a right packet decapsulation.
- Specific base station and node module parameters: contains values for the specific base station and node constraints. For instance, the base station position is fixed and it doesn't send any data. Therefore, its application and network layer will be simpler than the node layers. The nodes have additional features and constraints like constant mobility and battery consumption. Furthermore, they perform data generation and use specific routing protocols.

Besides the *General* section, the file can contain other sections. These sections would contain specific values to perform different simulations and tests. As a result of this, the network behavior after the modification of some significant parameters can be studied in an easier way. For instance, the **omnet.ini** file of this simulation contains some other sections to study the battery consumption by changing some parameters like the sensing time, the data size or the nodes speed.

The next sections to the *General* section are declared as follows:

```
[Config SpecificConfigName]
```

[4] These subsections, not required by the simulation environment, establish an internal file structure.

These sections don't need to specify all the parameters like in the *General* section, but only the specific parameters of the simulation. Thus, the non-specified parameters will be taken from the *General* section and, therefore, the desired behavior with a shorter and better **omnet.ini** file structure will be managed.

As an example, these sections are used in the automation of the simulations to obtain the simulation results. In the **omnetpp.ini** used in this project, after the *General* section, appear diverse sections for the different simulation executions.

```
[Config Packet256sensing30s]
output-scalar-file = ${resultdir}/256bits/sensing=30s/v=${runnumber}s.sca
output-vector-file = ${resultdir}/256bits/sensing=30s/v=${runnumber}s.vec
baseSim.*.appl.headerLength = 256bit
baseSim.node[*].appl.sensingTime = 30
baseSim.node[*].mobility.speed = ${0, 1, 2, 4}mps

[Config Packet256sensing60s]
...

[Config Packet1024sensing120s]
...
```

All these sections, for a fixed data size and time between two sensing acts, carry out the execution with different node speed values.

5.4 Conclusions

Making a good description of the simulation scenarios is a labor which helps researchers and developers to understand the problem basis and circumstances and get a fast problem's knowledge.

The design and implementation of the simulation scenarios follows the network construction architecture recommended by MiXiM. This fact helps to make use of the offered reusability principles, obtain a faster and more robust implementation and understand in a faster manner the network architecture for any future studio or development over it. Therefore, making a correct design and implementation, the network's behavior monitoring task detailed on the next chapter will be simpler and easier.

6 Evaluation of the simulation scenarios

6.1 Introduction

The simulations of the described scenarios will describe how the network behaves. Therefore, it is very important to analyze which factors or parameters will provide a general idea about the protocol operation and which will also provide the specific values that will allow carrying out an adequate analysis of results.

After choosing the parameters to analyze, it can be useful to hazard which kind of results will be obtained and why these results will be obtained. This fact will help to get an approximation how will operate the network and how will the results look like.

6.2 Radio model

For the current simulation, a simple radio model has been taken from [24]. The described model in the referred source assumes energy dissipation of E_{elec} = 50 nJ/bit to run the transmitter or receiver circuitry and E_{amp} = 100pJ/bit/m^2 for the transmit amplifier to achieve an acceptable E_b/N_o (Signal to Noise Ratio).

Table 3. Radio characteristics

Operation	Energy dissipated
Transmitter Electronics($E_{Tx\text{-}elec}$) Receiver Electronics ($E_{Rx\text{-}elec}$) ($E_{Tx\text{-}elec} = E_{Rx\text{-}elec} = E_{elec}$)	50nJ/bit
Transmit Amplifier (E_{amp})	100pJ/bit/m^2

It is also assumed an r^2 energy loss due to the channel transmission. Thus, to transmit and receive a k-bit message a distance d using this radio model, the radio expends:

$$E_{Tx}(k, d) = E_{elec} * k + E_{amp} * k * d^2$$

$$E_{Rx}(k) = E_{elec} * k$$

The value of these parameters makes the message transmission and reception not low cost operations. Therefore, the protocols should minimize the number of transmit and receive operations by means of switching its state between active and sleep (or even idle) when required in order to minimize the energy consumption.

In order to simplify the model and measure only the routing protocol energy consumption, it is assumed that the rest of the modules like the mobility or the data sensing and processing module don't have any energy consumption.

6.3 Monitoring the network's behavior

The quality of the obtained results will depend on the network aspects to monitor. For this research, it results very important to know how is carried out the energy consumption in the simulated network, and also why is dissipated such amount of energy. Therefore, it has special interest to monitor the following network aspects:

- Number and percentage of alive nodes during the network lifetime: these characteristics will show when the nodes die, and hence, how many nodes are alive during the network lifetime. This will illustrate the routing protocol's energy efficiency.
- Number of transmissions by each node: this attribute will indicate the amount of data sent by nodes during their lifetime, and thus, the network's balance. For instance, if there is a big difference between the total transmissions of two network nodes (bigger than 2 times), it would mean that the difference between the two nodes lifetime is significant. Therefore, if the nodes with the lower lifetime are located in a nearby distance between them, the data sensing from these areas can't be obtained.
- Node lifetime: this parameter will show how long will be the node, i.e., how much time will remain alive. This parameter is useful to do diverse comparisons, for example, the difference when the first or the last node dies between two protocols, even the difference between when the first and the last node dies into the same protocol.

6.4 Expected results

6.4.1 General results

The simulation results will depend on the three parameters mentioned in Section 7.2 and the Appendix I. Table of Specifications of the simulation scenarios: the speed of the nodes, the size of the generated data and the frequency of the generated data.

Amongst the simulation variables mentioned above, the most decisive parameter into the simulation results will be the network node speed. This parameter introduces two completely different scenarios in terms of node mobility:

- Static node network: when the value of this parameter is 0, it indicates that the network nodes are static. In a large scope WSN, there will be a big difference between nodes' lifetime depending on the distance to the base station. I.e., the nodes located far from the base stations will spend a large amount of energy in their transmission, whereas the nodes near to the base station will spend a lower amount of energy in the transmissions and, therefore, their lifetime will be much longer. Therefore, there will be a big difference between when the first node and the last node dies. With the use of different routing protocols, two different scenarios and behaviors will be obtained:
    - With **Direct Transmission** protocol, the simulation results should show that the number of alive nodes decreases in an exponential manner due to the fact that the node distance to the base station influences quadratically into the energy consumption.

  - With **LEACH** protocol, the simulation results also should show that the number of alive nodes decreases in an exponential manner, but with a lower slope. This is because of in each round, only the cluster head nodes transmit to the base station. As the cluster head nodes store all the cluster members' messages, the cluster head nodes located far from the base station will die much earlier than those located near to the base station.

- Dynamic node network: when the value of the *speed* parameter is greater than 0, the network nodes have constant movement. When the speed value is increased, the nodes tend to travel across the entire simulation field. Hence two new scenarios different from the static node network are obtained again:
  - With **Direct Transmission** protocol, the nodes' lifetime will become closer to each other because they transmit messages to the base station from different places. This fact involves that also the moment when the first node dies and the last node dies will become closer.
  - With **LEACH** protocol, the node's lifetime will also become closer, but in a smaller manner than in Direct Transmission. This is because only the cluster head nodes make large distance transmissions to the base station. The graphical representation of the results of the number of alive nodes in LEACH should show a similar shape than the Direct Transmission results, but with a lower slope.

With the other two variables, associated results should be found in the case of the Direct Transmission. Associated results should also be found with LEACH when the sensing time is equal to the LEACH round time. This is due to the fact that similar quantities of data are generated. For instance, we have two different scenarios: in the scenario A, the sensing time is fixed to 30 seconds and the data size to 512 bits. In the scenario B, the sensing time is fixed to 60 seconds and the data size to 256 bits. In both cases, the amount of data generated per minute is similar (512 bits per min.). Next is presented a table with the results association of the simulation tests:

Table 4. Relation between results with different simulation scenarios

Data quantity (in bits per min)	Time between sensing acts (in seconds)	Data size (in bits)
256	60	256
	120	512
512	30	256
	60	512
	120	1024
1024	30	512
	60	1024

Thus, only small differences will be found in the energy consumption between these associated results because of two factors:

- The size of the packet headers: with lower data sizes, the headers overload is bigger, i.e., *data size/headers size* ratio is increased. Hence, the overload requires extra energy consumption.
- The energy consumption of the transmitter circuitry: if data packets are generated and transmitted in more often manner, when the transmission runs the transmitter circuitry, it will spend more energy in the same proportion.

On the other hand, the frequency of the data generation will depend on the application scenario. Therefore, the best relation between generated data and data size should be pursued.

6.4.2 Comparison between Direct Transmission and LEACH results

In a general manner, could be affirmed that the network with the LEACH routing protocol will present a better behavior than the network with the Direct Transmission routing protocol in terms of energy consumption. In some surveys as [24] and [48], is demonstrated that the communication energy with LEACH is a few times lower than the communication energy with Direct Transmission, and the moment when the first and the last node dies is also a few times later.

Due to the simulation conditions are different; the obtained results could differ from the obtained results in the above referred surveys. However, the energy consumption with LEACH protocol should be lower because of diverse aspects:

- The large distance transmissions to the base station are only made by the cluster head nodes. Hence, the energy consumption of the cluster members is reduced.
- The data aggregation carried out by the cluster head node reduces the total amount of energy transmitted to the base station and, therefore, the total energy consumption per round.

6.5 Conclusions

In order to make a proper evaluation of the simulated network, it is very important to study which factors will affect to the network behavior. Due to the WSN have battery constraints; a good specification of the radio model and the energy consumption conditions is essential for the developed study.

After comprehending that some factors such as the speed of the nodes or the quantity of the data generated are significant to the network results, it would be interesting to imagine how the variation of these parameters will affect the selected monitoring parameters as the number of alive nodes, node lifetime and number of transmissions per node. By means of the value of these parameters, the energy consumption and the relation between the obtained results can be analyzed. The analysis should show that the speed of the nodes affects completely to the energy consumption and, in general, LEACH obtains better results than Direct Transmission in terms of less energy consumption.

7 Simulation results

7.1 Introduction

The simulations results are the objective evidences which demonstrates how is the operation of the implemented network and how the consumption of resources happens. Hence, a correct analysis of the obtained results with graphical representations will allow making some comparisons, extracting adequate conclusions and anticipating a preliminary search of improvements.

Even though there hasn't been possible to show simulation results about the LEACH protocol behavior, the general and specific problems are enumerated, in addition to the following steps into future versions of this project. This description will help to prevent from experiencing the same suffered problems.

The Direct Transmission results will show its behavior depending on the data size and the interval of data generation. The mobility conditions will also demonstrate how significant the existence of node mobility into the energy consumption is.

7.2 Parameters of the simulation tests

In order to obtain multiples results, a series of test has been prepared for the different routing protocols described above. This series of test will determine how the network behaves with a specific routing protocol and fixed parameters. In consequence, the values which demonstrate the best network behavior for each protocol can be obtained. Furthermore, it can be compare how influence the same values into the different protocols. Next on Figure 30 is showed a schema with the diverse test for different values and the obtained network behavior.

Data size = 256 bits		Sensing time		
		30 [s]	*60 [s]*	*120 [s]*
Node speed	*0 [m/s]*	B_1	B_2	B_3
	1 [m			
	2 [m			
	4 [m			

Data size = 512 bits		Sensing time		
		30 [s]	*60 [s]*	*120 [s]*
Node speed	*0 [m/s]*	B_{13}	B_{14}	B_{15}
	1 [n			
	2 [n			
	4 [n			

Data size = 1024 bits		Sensing time		
		30 [s]	*60 [s]*	*120 [s]*
Node speed	*0 [m/s]*	B_{25}	B_{26}	B_{27}
	1 [m/s]	B_{28}	B_{29}	B_{30}
	2 [m/s]	B_{31}	B_{32}	B_{33}
	4 [m/s]	B_{34}	B_{35}	B_{36}

Figure 30. Schema of the simulation tests

The simulation variables and its values are described below:

- **Node speed:** it fixes the speed of the nodes in the simulation field, in meters per second. The nodes will describe a constant and linear movement and, when the area limit is reached, it fixes a new direction. The values for the tests will be: 0 (static nodes), 1, 2 and 4 m/s.
- **Sensing time:** it defines the time between two consecutive sensing acts, in seconds. This value determines when a data packet is generated, but not when it is sent to the base station. For instance, in LEACH protocol, the time between different transmissions is fixed by the round time. The values for the tests will be: 30, 60 and 120 s.
- **Data size:** it determines the size of the data sent to the base station, in bits. The bigger data size, the larger amount of data sent to the base station. It can be result obvious the fact that a bigger data size will offer worse results, but, for example, in several scenarios could be more interesting to increase the data size and also the time between two sensing acts to obtain less energy consumption. The values for the tests will be: 256, 512 and 1024 bits of data size.

7.3 Direct Transmission results

The simulation results of Direct Transmission protocol obtained from the execution of the simulation scenarios are shown during this subsection.

The first analysis showed on Figure 31 consists on the study of the tendency of the energy consumption by means of the number of alive nodes with a fixed data size of 1024 bits and variability in the interval of data generation and the speed of the nodes.

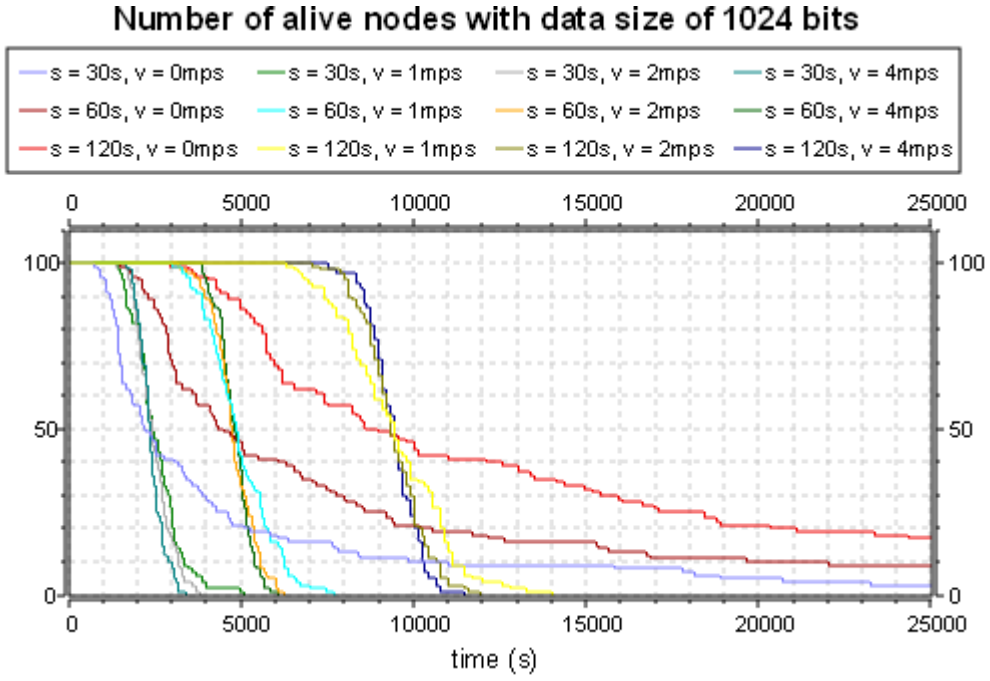

Figure 31. Number of alive nodes with data size of 1024 bits

The chart showed above on Figure 31 demonstrates that there is a big difference between the static and the dynamic simulation scenario in terms of node mobility. Into

the chart can be distinguished very clearly four different tendencies corresponding to the interval of data generation. Within these four tendencies, significant differences cannot be appreciated except when the speed of the nodes is 0. In these cases, the shape of the number of alive nodes curve is decreasing exponential. The meaning of these tendencies consists on the fact that, after the moment of the death of the first node, the rest of the nodes will continue dying in a decreasing exponential manner, i.e., the number of alive nodes will decrease more quickly after the death of the first node, but during the execution time, the number of alive node will decrease in a more and more slowly manner. For the rest of the tendencies with a not null mobility value, the tendency showed is quasi-linear. The meaning of these tendencies consists on the fact that, after the moment of the death of the first node, the nodes from the simulated network continue dying in a quasi-uniform manner.

The next analysis consists on how much affects the interval of data generation and the data size for a fixed speed of the nodes.

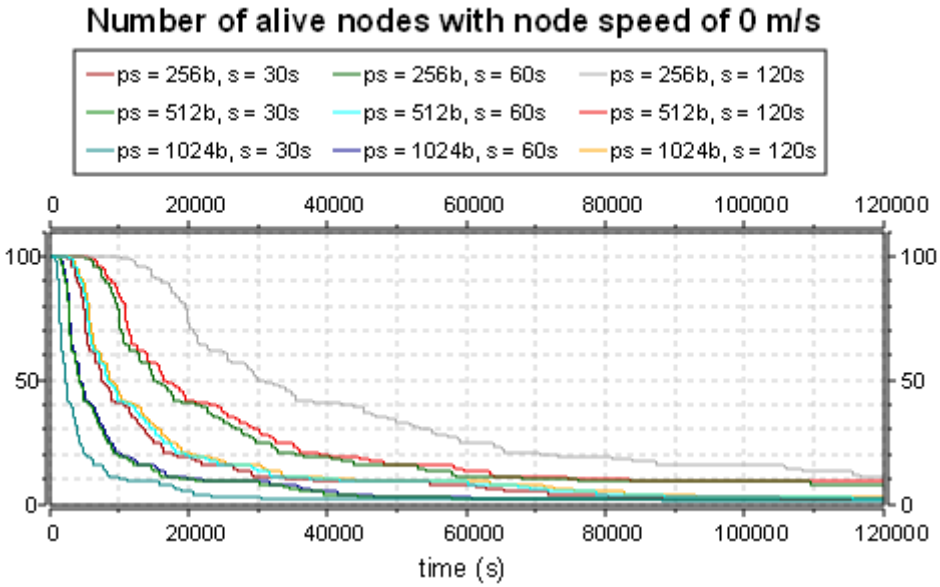

Figure 32. Number of alive nodes with node speed of 0 m/s

The chart showed on Figure 32 illustrates the behavior of the network in different scenarios for a node speed of 0 meters per second. Into this chart, the shape of the curve of the energy consumption when the network is static can be distinguished clearly. Furthermore, three zones with overlapped curves can be appreciated. These zones are corresponded with the similar quantities of data generated per minute during the simulation and mentioned in Section 6.4.1. These similar zones demonstrate that the node energy consumption is proportional to the amount of data generated per unit of time. Furthermore, it can be seen that the number of alive nodes during the time is almost duplicated when the amount data generated per unit of time is reduced to the half part, and also the number of alive nodes during the time is almost quadruplicated when the amount data generated per unit of time is reduced a fourth of the reference value.

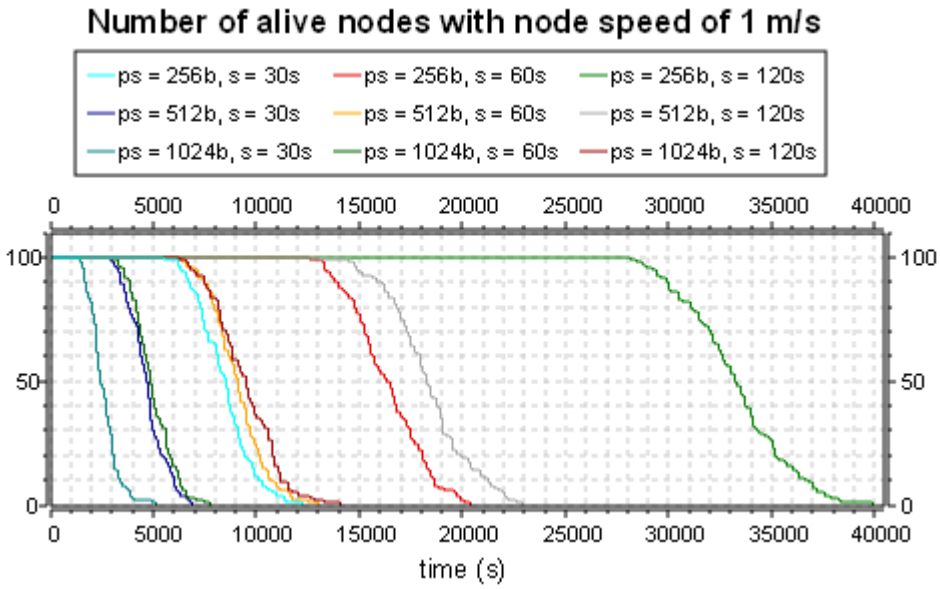

Figure 33. Number of alive nodes with node speed of 1 m/s

The chart showed on Figure 33 illustrates the behavior of the network in different scenarios for a node speed of 1 meter per second. The shape of the curves within this chart differs notably from the shape of the curves showed on Figure 32. This difference comes directly from the mobility of the network as it was described above. This chart distinguishes once again three zones where the curves are overlapped, and they correspond with the quantity of data generated per minute during the simulation, with similar interpretation.

Analyzing the differences between the two last charts, it is easy to recognize that the network lifetime in the static network is much longer than the network lifetime in the dynamic network. The comparison of the values shows that the network lifetime in a static network is around 50 times longer than in a network with node mobility of 1m/s for different data size and interval of data generation values as showed in Table 5. Nevertheless, in the dynamic network can be appreciated that the nodes die in a more uniform manner, whereas in the static network, the nodes die from the further zones to the closer zones to the base station. Furthermore, the moment when the first node dies into the dynamic scenarios is greater than two times than the value in comparison with the static scenarios is showed on Table 5. A similar behavior has been observed with speed values of 2 and 4 meters per second, and the rest of data size and intervals of data generation scenarios.

Table 5. First and last node dies values with data size of 256 bits

	Data size = 256 bits					
	sensing_time = 30 s		**sensing_time = 60 s**		**sensing_time = 120 s**	
	v = 0 m/s	*v = 1 m/s*	*v = 0 m/s*	*v = 1 m/s*	*v = 0 m/s*	*v = 1 m/s*
First node dies (in s)	2570.22	5450.31	5135.22	12578.34	10265.22	28146.71
Last node dies (in s)	560088.84	12172.21	1120173.84	20313.88	2240343.84	39902.11

Finally, the last two analyses have the purpose of demonstrate the drawbacks of the static scenarios in terms of energy balance in comparison with dynamic scenarios.

The chart showed on Figure 34 illustrates the number of transmissions to the base station for each node for a data size of 1024 bits and an interval of data generation of 30 seconds. As it can be appreciated, in the static network scenario, there is a big difference between the number of transmissions of nodes located near to the base station and nodes located far from the base station. Hence, even though the network lifetime is decreased, the node mobility influences positively to the network energy consumption balance.

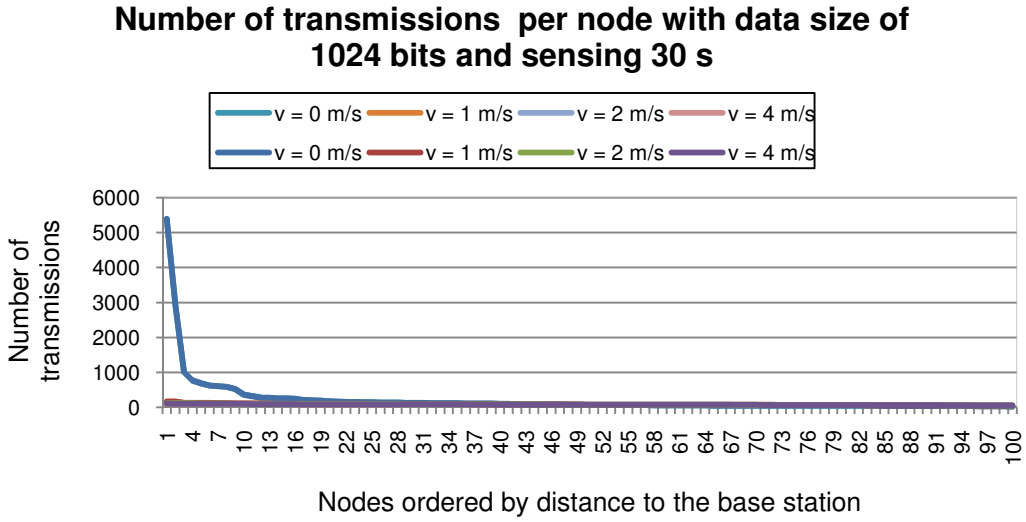

Figure 34. Number of transmissions per node with data size of 1024 bits and interval of data generation of 30 s

The last chart showed on Figure 35 illustrates how is distributed the energy consumption along the network. A notably difference between the tendency of the curve in the static scenario and the dynamic scenarios is appreciated. Whereas the dynamic scenarios show a relative small difference between the average energy per transmission of all nodes, the difference within the static scenario between the closest and the furthest node to the base station greater than 20 times.

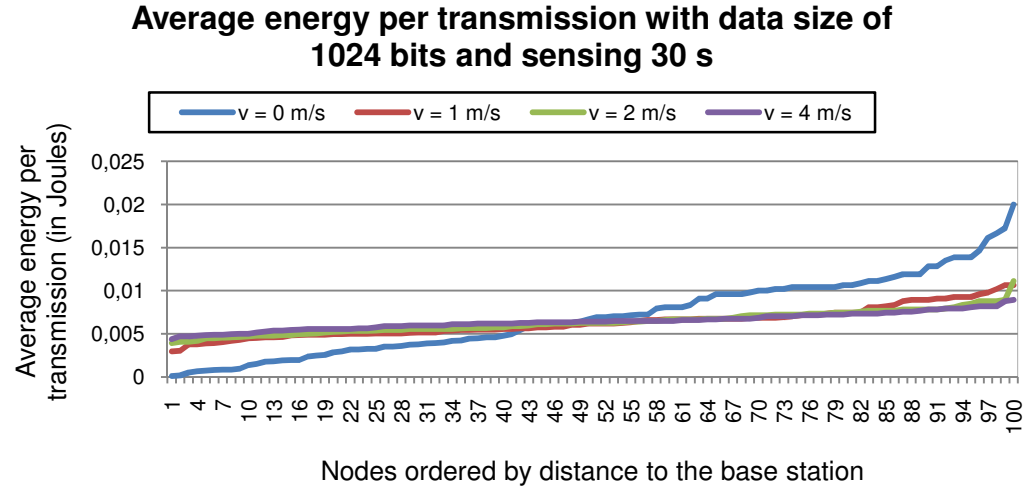

Figure 35. Number of transmissions per node with data size of 1024 bits and interval of data generation of 30 s

7.4 LEACH results

Owing to impossibility of finishing the LEACH protocol implementation, simulation results about the LEACH protocol operation couldn't be obtained. During this subsection, the general and the specific caused of the impossibility of finishing the LEACH implementation are detailed. In addition, the remaining work to obtain correct LEACH operation results and protocol improvements is described.

The general factors which hindered to finish successfully the research are presented below:

- Insufficient time: the project's schedule has been altered several times during the development of the project. The main cause has been the increment of the complexity because of the advent of new difficulties.
- Insufficient personnel: the lack of information and resources and the presence of bugs into the simulation environment have been one of the main reasons of the experienced delay. A research group consisting of two or three members would have helped to identify the experienced problems and provide some possible solutions.

The specific factors and problems that made the development of the project difficult are detailed below:

- Huge protocol variety to research: due to the fact that WSN are a new technology, an absence of protocol standards is present This fact difficult the analysis of the current MAC and network protocols and also complicates the election of the suitable protocols for the present study. Currently there exists numerous MAC protocols for WSN, but the WSN constraints have caused that there is a big quantity of basic routing protocols and innumerable lines of research about specific improvements of any routing protocol.
- Diversity of simulation environments: the variety of simulation environments, each one with their own features, but also with their own language syntax, made difficult to carry out a clear comparison between them. The general simulation environments required a specific framework to obtain a WSN simulation because they didn't support the simulation of WSN in native manner, but these frameworks are relatively new and specific of specific researches. Therefore, outside from the framework scope, a lack of unity between modules and presence of bugs has been found. On the other hand, the specific simulation tools, which in include the simulation of the node operating system, or even emulation of the memory mapping, would have added an unnecessary complexity to the current research.
- Instability of the simulation environment: several bugs and stability problems were experienced. A major bug[1] was found and reported to the project's direction. Furthermore, numerous simulation environment crashes happened during the architecture node construction, code implementation, debugging and

[1] Reported bug: imposible to make a 2nd extend into the OMNeT++ hierarchy. Impossible to extend from BaseNetwLayer class, which extends from BaseLayer class. Solution: Ext2NetwLayer was modified to extend from BaseLayer. Further information can be found in: http://dev.omnetpp.org/bugs/view.php?id=88

simulation runs were experienced. These facts hampered the progressive project development and, therefore, retarded and hindered the LEACH results achieving.

- Absence of ready-to-use protocol implementations: the lack of ready-to-use protocols has remarkably reduced the flexibility and the possibility of testing different options within the simulation environment. The whole implementation of the network architecture and the used protocols has been needed, and debugging, specific corrections and protocol operation tests have been required.
- Complexity of the simulation architecture: the module and class hierarchy is so extensive and interrelated that makes extremely difficult the debugging process. The process of sending an airframe requires the use of numerous and interconnected classes from the nodes, the channel and the simulation environment. This fact, together with the environment instability and the bugs found into the used framework made extremely complex the process of finding the origin of the obtained errors during the execution.
- Lack of documentation into the API of the used framework: the procedure of looking up into the used framework's API showed that there is a lack of description in several classes and subroutines. This fact, together with the complexity of the simulation architecture described above, made difficult to understand the simulation operations and more difficult the search of specific functionality
- Bugs found into the used framework: several implementation bugs where found into the framework implementation. The first bug[2] made an error into the network address resolution through the MAC address. The second bug[3], with mayor severity, blocked the energy draw process and hindered the node energy consumption. The process of finding and solving the errors contributed to the simulation development delay.

As it has been expounded in Section 4.3.2, the full LEACH protocol operation has been implemented. Therefore, the remaining steps to finish the whole research are explained below:

- Check the LEACH protocol correct behavior: it is necessary to check the correct behavior of the implemented protocol by means of error debugging, modification of operation problems derived from protocol understanding difficulties and protocol operation tests execution.
- Results achieving: the next step consists on running the set of tests in order to obtain data results and prepare descriptive graphical representations of the obtained results.

[2] Reported bug: wrong information given by the `getNetwAddr(const int macAddr)` subroutine, line 54, within the BaseArp.cc class:
http://mixim.sourceforge.net/doc/doxy/a00005.html

[3] Reported bug: implementation error into the `DrawAmount` constructor class made the value initialization impossible:
http://mixim.sourceforge.net/doc/doxy/a00062.html

- Analysis and comparison of results: the following step consists on making a correct analysis with the obtained simulation results and extracting some conclusions about the protocol operation.
- Design ad implementation of power management improvements: after having results about how the LEACH protocol behaves with specific values should happen one important step: to plan the design and the implementation of a LEACH energy consumption improvement from the obtained results.
- Comparison and measurement of results: the last step will consist on comparing the results obtained from the implemented protocol improvement with the base protocol and measure with specific values how important has been the power management improvement.

7.5 Conclusions

During this chapter has been reflected how the mobility of the network, besides the network lifetime is reduced, delays the moment when the first node dies and also introduces uniformity during the node depletion process with Direct Transmission protocol. This fact represents that the network energy consumption is more balanced than in a static network scenario and, therefore, the nodes will die in a regular manner and not from the further places to the closer zones to the base station. Thus, the simulated network won't have any place in the simulation field without sensing data while the whole network keeps alive.

Although simulation results from the LEACH protocol couldn't be obtained, the experienced problems suffered during the project development have been reflected in order to avoid these problems during future revisions and the guidelines for further versions have been specified.

8 Conclusions

In the current document, the implementation of a WSN architecture in a simulation environment has been carried out. This implementation will evaluate different simulation scenarios of large scope clustered-type networks. The purpose of the implementation consists on the research and the obtaining of a power management improvement over LEACH, one of the present WSN network protocols. The search of the power management improvement will be performed through the analysis of the network behavior. This network will be obtained by means of simulation executions after the variation of values of specific parameters characteristic of the WSN operation.

As a result of the WSN communication architecture, their design factors and requirements, the state of the art of the network and Medium Access Control WSN protocols, and the properties of the simulation scenarios, the election of the MAC and routing protocols has been carried out according to the features and the constraints of WSNs, as well as the features of the simulation scenarios. The analysis about the WSN protocols carried out showed that CSMA, with its simplicity, its well-know behavior and adequate performance; and LEACH, with its clustered type organization, the balance of the overall energy consumption that carries out and its relevance on the current researches, become suitable protocols suitable for the carried out research.

The election of the simulation scenario was carried out in regard to compliance with the necessary requirements for the implementation of a WSN architecture, as well as other features which facilitate the implementation development. The analysis carried out about the most common WSN simulation environments showed that OMNeT++, together with MiXiM development framework, meet the majority of the simulation requirements thanks to the simulation structure, the WSN simulation capabilities and the graphical runtime environment, which facilitates the design, debugging and test of large scope wireless sensor networks.

The evaluation of the simulation scenarios has been specified with the highest detail as possible. This fact will help the obtaining of simulation results and the reproduction of the tests in other simulation environments. These simulation tests showed about Direct Transmission protocol how the mobility of the network, besides the network lifetime is reduced, delays the moment when the first node dies and also introduces uniformity in during the node depletion process, providing in that manner a more balanced overall energy consumption. Besides it couldn’t be possible to obtain simulation results about the LEACH protocol operation, the experienced problems during the project development have been reflected and the guidelines for further versions have been specified.

Therefore, it can be determined that the WSN architecture implemented perform the obtaining of desired results during the development of this project, and could serve as base implementation for future versions of the developed work.

9 Future work

The guidelines for finishing all the objectives from the current study have been detailed in Section 7.4. After obtaining the results about the LEACH protocol operation, the design and implementation of power management improvements over the LEACH operation can be carried out through different alternatives or possibilities. Multiples lines of research are about energy consumption improvements over LEACH have been made and are also currently open. Some possibilities are introduced in [24]:

- Energy-aware threshold: the inclusion of an energy level parameter into the calculation of the threshold during the Advertisement Phase will enable the election of the cluster head nodes in relation to the amount of energy of the nodes scattered in the simulation field. Thus, the overall energy consumption of the network would be more balanced.
- Hierarchical clustering: the LEACH version implemented in this project can be extended to form hierarchical clusters. In this manner, a hierarchy could be built where the cluster head nodes would communicate with "super-cluster head" nodes and so until the top layer of the hierarchy, at which point the data would be sent to the base station. This architecture could save tremendous amount of energy in large networks WSN as the network scenario of the current project.

Furthermore, there are other multiple different possibilities for the research of energy consumption improvements. Some possibilities are described below:

- Better integration between MAC and network protocols: the MAC protocols described in this document provide different features. The correct integration of other MAC protocols with LEACH could provide a notably decrement of the overall energy consumption. Possible examples could be WiseMAC, which reduces the energy consumption, or TRAMA, which increases the sleeping mode time percentage and decrements the collision probability in comparison with CSMA based protocols,
- Modification of the LEACH operation: some modifications over the LEACH protocol operation could provide better results in terms of overall energy consumption. One possible modification is described below.

<<At the beginning of each LEACH round, the base station broadcasts a "round starts" message to the entire network. The nodes that have been elected themselves cluster head nodes broadcast its "cluster head status" message. Furthermore, in this message, they attach the signal strength of the base station broadcast that they received. At this point, if any other cluster head node listen the "cluster head status" message, it compares its base station received signal strength with the signal strength attached in the cluster head node message. If the value in the "cluster head status" message is smaller, it stores the source of the message as "router node". When the Data Transmission Phase begins and the cluster head nodes have received all the messages from the cluster members, if the cluster head nodes have stored any "router node", they will send their data to this node instead of sending the message directly to the base station>>.

Although can be thought that this scenario could show the "hot spot" problem, can be considered that the network mobility can solve this problem. Furthermore, this new functionality can be disabled when the node energy level reaches a specific threshold.

10 References

1. **Sohraby, Kazem, Minoli, Daniel and Znati, Taieb.** *Wireless Sensor Networks: Technology, Protocols, and Applications.* s.l. : Wiley-Interscience, 2007.

2. **Karl, Holger and Willig, Andreas.** *Protocols and Architectures for Wireless Sensor Networks.* 1. s.l. : Wiley, 2005.

3. *Energy-efficient wireless sensor network design and implementation for condition-based maintenance.* **Tiwari, Ankit, Ballal, Prasanna and Lewis, Frank L.** s.l. : ACM, 2007, ACM Trans. Sen. Netw., Vol. 3, p. 1.

4. *A Distributed Efficient Architecture for Wireless Sensor Networks.* **Lin, Chuan, et al.** 2007. Proc. 21st International Conference on Advanced Information Networking. Vol. 2, pp. 429-434.

5. *An energy-efficient MAC protocol for wireless sensor networks.* **Ye, Wei, Heidemann, J. and Estrin, D.** 2002. SenSys '03: Proceedings of the 1st international conference on Embedded. Vol. 3, pp. 1567-1576.

6. *An adaptive energy-efficient MAC protocol for wireless sensor networks.* **Dam, Tijs van and Langendoen, Koen.** s.l. : ACM, 2003. Proc. IEEE Twenty-First Annual Joint Conference of the IEEE Computer. pp. 171-180.

7. **networks, Medium access control with a dynamic duty cycle for sensor.** 2004. Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE. Vol. 3.

8. *Versatile low power media access for wireless sensor networks.* **Polastre, Joseph, Hill, Jason and Culler, David.** s.l. : ACM Press, 2004. SenSys '04: Proceedings of the 2nd international conference on Embedded. pp. 95-107.

9. *Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks.* **El-Hoiydi, Amre.** 2002. Proc. Seventh International Symposium on Computers and Communications. pp. 685-692.

10. *An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks.* **Lu, Gang, Krishnamachari, Bhaskar and Raghavendra, Cauligi S.** s.l. : IEEE Computer Society, 2004, Parallel and Distributed Processing Symposium, International, Vol. 13, p. 224a.

11. *Energy-Efficient, Collision-Free Medium Access Control for Wireless Sensor Networks.* **Rajendran, Venkatesh, Obraczka, Katia and Garcia-Luna-Aceves, J.** s.l. : Springer, 2006, Wireless Networks, Vol. 12, pp. 63-78.

12. *Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks}.* **Jamieson, Kyle and Balakrishnan, Hari and Tay, Y. C.** 2006. Wireless Sensor Networks. pp. 260--275.

13. *Carrier sense multiple-access modes and their throughput-delay characteristics.* **Kleinrock, Leonard, Fouad and Tobagi, A.** 1983, IEEE Trans. Comm, Vol. 23, pp. 1400-1416.

14. *A Survey of Gossiping and Broadcasting in Communication Networks.* **Hedetniemi, Hedetniemi and Liestman.** 1988, Networks: An International Journal, Vol. 18.

15. *Negotiation-based protocols for disseminating information in wireless sensor networks.* **Kulik, Joanna, Heinzelman, Wendi and Balakrishnan, Hari.** s.l. : Kluwer Academic Publishers, 2002, Wirel. Netw., Vol. 8, pp. 169-185.

16. *Directed diffusion: a scalable and robust communication paradigm for sensor networks.* **C., R. Govindan and Estrin, D.** 2000. Proceedings of the sixth annual international conference on Mobile computing and networking. pp. 56-67.

17. *Rumor Routing Algorithm For Sensor Networks.* **Estrin, David Braginsky and Deborah.** 2002. In Proceedings of the First Workshop on Sensor Networks and Applications (WSNA).

18. *Energy efficient routing in wireless sensor networks.* **Schurgers, C. and Srivastava, M.** s.l. : IEEE, 2001. Proceedings of MILCOM 2001. pp. 357-361.

19. *Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks.* **Chu, M., Haussecker, H. and Zhao, F.** 2002, International Journal on High Performance Computing Applications.

20. *Energy Aware Routing for Low Energy Ad Hoc Sensor Networks.* **Shah, Rahul C. and Rabaey, Jan M.** 2002. IEEE Wireless Communications and Networking Conference (WCNC).

21. *A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks.* **Ye, F., et al.** 2001. 10th IEEE International Conference on Computer Communications and Networks (ICCCN).

22. *The Cougar approach to in-network query processing in sensor networks.* **Yao, Yong and Gehrke, Johannes.** 2002, SIGMOD Record, Vol. 31.

23. *The ACQUIRE Mechanism for Efficient Querying in Sensor Networks.* **Sadagopan, Narayanan, Krishnamachari, Bhaskar and Helmy, Ahmed.** 2003. In IEEE International Workshop on Sensor Network Protocols and Applications (SNPA'03). pp. 149-155.

24. *Energy-Efficient Communication Protocol for Wireless Microsensor Networks.* **Heinzelman, Wendi Rabiner, Chandrakasan, Anantha and Balakrishnan, Hari.** s.l. : IEEE Computer Society, 2000. p. 8020.

25. *Proactive Context-Aware Sensor Networks.* **Ahn, Sungjin and Kim, Daeyoung.** 2006. EWSN. pp. 38-53.

26. *PEGASIS: Power-efficient gathering in sensor information systems.* **Lindsey, S. and Raghavendra, C. S.** 2002. Aerospace Conference Proceedings, 2002. IEEE. Vol. 3, pp. 3-1125--3-1130 vol.3.

27. *TEEN: a routing protocol for enhanced efficiency in wireless sensor networks.* **Manjeshwar, A. and Agrawal, D. P.** 2001. Parallel and Distributed Processing Symposium., Proceedings 15th International. pp. 2009-2015.

28. *APTEEN: a hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks.* **Manjeshwar, A. and Agrawal, D. P.** 2002.

Parallel and Distributed Processing Symposium., Proceedings International, IPDPS 2002. pp. 195-202.

29. *Minimum energy mobile wireless networks.* **Rodoplu, V. and Meng, T. H.** 1999, Selected Areas in Communications, IEEE Journal on, Vol. 17, pp. 1333-1344.

30. *Routing techniques in wireless sensor networks: a survey.* **Al-Karaki, J. N. and Kamal, A. E.** 2004, IEEE Wireless Communications, Vol. 11, pp. 6-28.

31. *Geography-informed energy conservation for Ad Hoc routing.* **Xu, Ya, Heidemann, John and Estrin, Deborah.** s.l. : ACM, 2001. MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking. pp. 70-84.

32. **Yu, Yan, Govindan, Ramesh and Estrin, Deborah.** *Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks.* UCLA Computer Science Department, Technical Report UCLA/CSD-TR-01-0023. 2001.

33. *GEDIR: Loop-free location based routing in wireless networks.* **Stojmenovic, Ivan and Lin, Xu.** 1999. IASTED International Conference on Parallel and Distributed Computing and Systems. pp. 1025-1028.

34. *Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks.* **Chen, Benjie, et al.** 2001. Proceedings of the 7th ACM International Conference on Mobile Computing and Networking. pp. 85-96.

35. *Protocols for self-organization of a wireless sensor network.* **Sohrabi, K., et al.** 2000, IEEE Personal Communications, Vol. 7, pp. 16-27.

36. *SPEED: a stateless protocol for real-time communication in sensor networks.* **He, Tian, et al.** 2003. Proc. 23rd International Conference on Distributed Computing Systems. pp. 46-55.

37. *Simulation tools for wireless sensor networks.* **E. Egea-López, J. Vales-Alonso, A. Martínez-Sala, P. Pavón-Marñio, and J. García-Haro.** Philadelphia, Pa, USA. : s.n., 2005. Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '05).

38. The Network Simulator, NS–2. [Online] http://www.isi.edu/nsnam/ns/.

39. OMNET++ discrete event simulator. [Online] http://www.omnetpp.org/.

40. *TOSSIM: accurate and scalable simulation of entire TinyOS applications.* **Levis, Philip, et al.** s.l. : ACM, 2003. SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems. pp. 126-137.

41. *Simulation of TinyOS Wireless Sensor Networks Using OPNET.* **Sumorok, Daniel, Starobinski, David and Trachtenberg, Ari.** 2008. OPNETWORK 2004.

42. Ptolemy II. Heterogeneous model and design. [Online]
http://ptolemy.berkeley.edu/ptolemyII/.

43. *Modeling of sensor nets in Ptolemy II.* **Baldwin, Philip, et al.** 2004. IPSN'04. pp. 359-368.

44. OMNeT++ User Manual. [Online]
http://www.omnetpp.org/doc/omnetpp40/Manual.pdf.

45. OMNeT++ IDE User Guide. [Online]
http://www.omnetpp.org/doc/omnetpp40/UserGuide.pdf.

46. MiXiM (mixed simulator) for OMNeT++. [Online]
http://sourceforge.net/apps/trac/mixim/

47. *Simulating Wireless Sensor Networks with OMNeT++.* **Mallanda, C., et al.** IEEE Computer, 2005.

48. *An application-specific protocol architecture for wireless microsensor networks.* **Chandrakasan, Anantha P., et al.** 2002, IEEE Transactions on Wireless Communications, Vol. 1, pp. 660-670.

11 Appendix I. Table of Specifications of the simulation scenarios

General parameters	
Playground size (in meters)	1000 × 1000
Number of nodes	100
Channel parameters	
Signal attenuation threshold (in dB)	-91
Minimum path loss coefficient	3.0
Carrier frequency of the channel (in Hz)	2.412×10^9
Radio model	
Transmitter energy consumption (in J/bit)	50×10^{-9}
Receiver energy consumption (in J/bit)	50×10^{-9}
Amplifier energy consumption (in J/bit/m^2)	0.1×10^{-9}
Physical layer parameters	
Strength of the thermal noise (in dBm)	-100
Sensivity (in dBm)	89
Switch times (in seconds)	
Rx to Tx	0.00012
Rx to sleep	0.000031
Tx to Rx	0.00012
Tx to sleep	0.000032
Sleep to Rx	0.000102
Sleep to Tx	0.000203
MAC layer parameters	
Queue length	5
Header length (in bits)	24
Slot duration (in seconds)	0.04
Difs time (in seconds)	0.0005
Maximum number of transmission attempts	14
Bit rate (in bps)	15360
Mobility module parameters	
Base station position ([x, y], in meters)	[10, 10]
Node position	Random
Battery module parameters	
Capacity (in J)	5.0

APPENDIX I. TABLE OF SPECIFICATIONS OF THE SIMULATION SCENARIOS

LEACH protocol parameters	
Round time	Value equal to *interval of data generation*
Slot time (in seconds)	0.08
Compression index	0.15
Waiting time[1] (in seconds)	0.02
Maximum cluster size	25 nodes

Variable simulation parameters	
Interval of data generation (in seconds)	{30, 60, 120}
Node speed	{0, 1, 2, 4}
Data size (in bits)	{256, 512, 1024}

[1] Amount of time that nodes should wait to switch to the next protocol phase.