UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UHA UNIVERSITE DE HAUTE-ALSACE Mulhouse-Colmar

DEPARTAMENTO DE COMUNICACIONES

IUT COLMAR

A GROUP-BASED ARCHITECTURE AND PROTOCOL FOR WIRELESS SENSOR NETWORKS

by

MIGUEL GARCIA PINEDA

Advisors:

PROF. DR. JAIME LLORET MAURI (UPV)

PROF. DR. PASCAL LORENZ (UHA)

January 2013

ABSTRACT

There are many works related to wireless sensor networks (WSNs) where authors present new protocols with better or enhanced features, others just compare their performance or present an application, but this work tries to provide a different perspective. Why don't we see the network as a whole and split it into groups to give better network performance regardless of the routing protocol?

For this reason, in this thesis we demonstrate through simulations that node's grouping feature in WSN improves the network's behavior. We propose the creation of a group-based architecture, where nodes have the same functionality within the network. Each group has a head node, which defines the area in which the nodes of such group are located. Each node has a unique node identifier (*nodeID*). First group's node makes a group identifier (*groupID*).

New nodes will know their *groupID* and *nodeID* of their neighbors. End nodes are, physically, the nodes that define a group. When there is an event on a node, this event is sent to all nodes in its group in order to take an appropriate action. End nodes have connections to other end nodes of neighboring groups and they will be used to send data to other groups or to receive information from other groups and to distribute it within their group. Links between end nodes of different groups are established mainly depending on their position, but if there are multiple possibilities, neighbor nodes could be selected based on their ability λ, being λ a choice parameter taking into account several network and nodes parameters. In order to set group's boundaries, we can consider two options, namely: i) limiting the group's diameter of a maximum number of hops, and ii) establishing boundaries of covered area.

In order to improve the proposed group-based architecture, we add collaboration between groups. A collaborative group-based network gives better performance to the group and to the whole system, thereby avoiding unnecessary message forwarding and additional overheads while saving energy. Grouping nodes also diminishes the average network delay while allowing scaling the network considerably. In order to offer an optimized monitoring process, and in order to offer the best reply in particular environments, group-based collaborative systems are needed. They will simplify the monitoring needs while offering direct control.

Finally, we propose a marine application where a variant of this group-based architecture could be applied and deployed.

RESUMEN

Existen muchos trabajos relacionados con las redes de sensores inalámbricas (WSNs), donde se presentan nuevos protocolos que aportan mejoras, otros trabajos donde se comparan para ver su rendimiento ó incluso presentan nuevas aplicaciones, pero este trabajo aporta otro punto de vista. ¿Por qué no ver la red como un conjunto que se divide en grupos para aportar un mejor rendimiento independientemente del protocolo de encaminamiento utilizado?

Para ello, en esta tesis, hemos demostrado a través de simulaciones, que la agrupación de nodos en WSNs aporta mejoras. Hemos propuesto la creación de una arquitectura basada en grupos, donde todos los nodos poseen la misma funcionalidad. Cada grupo tiene un nodo head que delimita la zona en la que estarán los nodos de un grupo. Cada nodo tiene un identificador único de nodo (*nodeID*) y el primer nodo de grupo creará un identificador de grupo (*groupID*).

Los nuevos nodos que se unan sabrán su *groupID* y el *nodeID* de sus vecinos. Los nodos end son, físicamente, los nodos que delimitan el grupo. Cuando hay un evento en un nodo, este evento se envía a todos los nodos de su grupo con el fin de tomar las medidas adecuadas. Los nodos end tienen conexiones con otros nodos frontera de los grupos vecinos y se utilizan para enviar/recibir información a/de otros grupos. Los enlaces entre los nodos end de diferentes grupos se establecen principalmente en función de su posición, pero si existen varias posibilidades, los vecinos se seleccionan en función de su capacidad λ, parámetro de elección que tiene en cuenta diversos parámetros de la red y de los nodos. Para establecer los límites del grupo, podemos considerar dos opciones: i) limitar el diámetro del grupo, y ii) establecer los límites de la zona cubierta.

Para mejorar la arquitectura propuesta añadimos colaboración entre grupos. Una red basada en grupos colaborativa da mayor rendimiento al grupo y a toda la red, evitando reenvío de mensajes innecesarios y por tanto un mayor ahorro de energía. El agrupamiento de nodos también hace disminuir el retraso medio de la red mientras permite que la red escale considerablemente. Para poder ofrecer un optimizado proceso de monitorización, y con el fin de obtener la mejor respuesta en determinados ambientes, son necesarios los sistemas colaborativos basados en grupos. Este tipo de sistemas simplificarán las necesidades de monitorización mientras aportan un control directo.

Por último, se propone una posible aplicación marítima donde podría implementarse una variante de esta arquitectura basada en grupos.

RESUM

Existeixen molts treballs relacionats amb les xarxes de sensors sense fils (WSNs), on es presenten nous protocols que aporten millores, altres treballs on es comparen per veure el seu rendiment o fins i tot presenten noves aplicacions, però aquest treball aporta un altre punt de vista. Per què no veure la xarxa com un conjunt que es divideix en grups per aportar un millor rendiment independentment del protocol d'encaminament utilitzat?

Per això, en aquesta tesi, demostrem a través de simulacions, que l'agrupació de nodes en WSNs aporta millores. Proposem la creació d'una arquitectura basada en grups, on tots els nodes tenen la mateixa funcionalitat. Cada grup té un node head que delimita la zona on hi estaran els nodes d'un grup. Cada node té un identificador únic de node (*nodeID*) i el primer node de grup crearà un identificador de grup (*groupID*).

Els nous nodes que s'uneixin sabran seva *groupID* i el *nodeID* dels seus veïns. Els nodes end són, físicament, els nodes que delimiten el grup. Quan hi ha un esdeveniment en un node, aquest esdeveniment s'envia a tots els nodes del seu grup per tal de prendre les mesures adequades. Els nodes end tenen connexions amb altres nodes end dels grups veïns i s'utilitzen per enviar/rebre informació a/d'altres grups. Els enllaços entre nodes end de diferents grups s'estableixen principalment en funció de la seva posició, però si hi ha diverses possibilitats, els veïns es seleccionen en funció de la seva capacitat λ, paràmetre d'elecció que té en compte diversos paràmetres de la xarxa i els nodes. Per establir els límits del grup, podem considerar dues opcions: i) limitar el diàmetre del grup, i ii) establir els límits de la zona coberta.

Per a millorar l'arquitectura proposada afegim col·laboració entre grups. Una xarxa basada en grups col·laborativa dóna major rendiment al grup i a tota la xarxa, evitant reenviament de missatges innecessaris i per tant un major estalvi d'energia. L'agrupament de nodes també fa disminuir el retard mitjà de la xarxa mentre permet que la xarxa escales considerablement. Per poder oferir un optimitzat procés de monitorització, i amb la finalitat d'obtenir la millor resposta a determinats ambients, són necessaris els sistemes col·laboratius basats en grups. Aquest tipus de sistemes simplificaran les necessitats de monitorització mentre aporten un control directe.

Finalment, es proposa una possible aplicació marítima on podria implementar-se una variant d'aquesta arquitectura basada en grups.

RESUME

Il existe de nombreux ouvrages liés aux réseaux de capteurs sans fils (WSNs), les auteurs présentent de nouvelles caractéristiques des protocoles de routage avec de meilleures performances et comparent ces dernières. Pourquoi ne pas diviser un réseau en groupes afin d'obtenir de meilleures performances quel que soit le protocole de routage?

Dans cette thèse, nous montrons par des simulations que les fonctionnalités de regroupement des nœuds de WSNs fournit des améliorations. Nous proposons donc la création d'une architecture fondée sur des groupes, où tous les nœuds ont la même fonctionnalité. Chaque groupe a un nœud head qui définit la zone dans laquelle sont regroupée les nœuds. Chaque nœud possède un identificateur de nœud unique (*nodeID*). Le premier groupe de nœuds crée un identifieur de groupe (*groupID*).

Les nouveaux nœuds connaissent le *groupID* et le *nodeID* de leurs voisins. Les nœuds end sont les nœuds qui définissent un groupe. Quand il y a un événement sur un nœud, cet événement est envoyé à tous les nœuds de son groupe afin de prendre les mesures appropriées. Les nœuds end sont connectés aux nœuds end d'autres groupes voisins et ils envoient/reçoivent des informations vers/à partir d'autres groupes. Les liens entre les nœuds end des différents groupes sont établis principalement en fonction de leur position, mais dans le cas où il y a de multiples possibilités, les nœuds voisins peuvent être choisis en fonction de leur capacité λ, qui prend en compte plusieurs paramètres du réseau. Afin de fixer des limites de groupe, nous pouvons envisager deux options: i) limiter le diamètre du groupe, et ii) établir les limites de la zone couverte.

Pour améliorer l'architecture proposée, nous allons ajouter des collaborations entre les groupes. Un réseau de collaboration des groupes offre une plus grande efficience le groupe et le réseau, en évitant le transfert des messages inutiles et donc économiser de l'énergie. Le regroupement des nœuds permet également de réduire le délai moyen de réseau, tout en permettant l'échelle du réseau considérablement. Afin d'assurer une surveillance des processus optimisés, et d'obtenir la meilleure réponse possible dans certains environnements, les systèmes sont tenus les groupes collaboratifs. Ces systèmes simplifiés alors que les besoins de surveillance fournissent un contrôle direct.

Enfin, nous proposons une application dans un contexte maritime qui pourrait être utilisée comme une variante de cette architecture de groupes.

DEDICATION

This Ph.D. dissertation is dedicated to a very special person who left us in February 2012, my “grandpa”. Thanks for everything that you gave me and taught me over these years, I would not have been the same without your advice.

I want you to know that every night I remember you. I wish you were here to share this moment with me but it is not possible. But I know, wherever you are, you will be proud of this work.

I LOVE YOU.

Este trabajo está dedicado a una persona muy especial para mí que nos dejó en Febrero de 2012. Gracias “iaio” por todo lo que me has aportado y enseñado a lo largo de todos estos años, posiblemente no hubiera sido el mismo sin tus consejos.

Quiero que sepas que todas las noches me acuerdo de ti. Me hubiera gustado que estuvieras compartiendo este momento conmigo pero no ha podido ser. Sé que haya donde estés estarás orgulloso de este trabajo.

TE QUIERO.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to take this opportunity to thank everybody who has supported me to do this Ph. D. thesis. In particular, I would like to show my gratitude to:

- Prof. Jaime Lloret for giving me the opportunity to start my research career with him and for his support in these years working together. Thanks for supervise my thesis.
- Prof. Pascal Lorenz for supervising my thesis in UHA. Thanks for your kindness for those months in Colmar. Everything was easier with his support.
- The members of the PhD committees, for their willingness to review the manuscript, for their positive feedback, and for the valuable suggestions they made, which enhanced the quality of this manuscript.
- All my colleagues at the Communications & Remote Sensing research laboratory and at the “Escuela Politécnica de Gandia – UPV” for their ideas on research topics, and for the personal and technical support.
- All my colleagues of Telecommunications’ Degree and Master.
- My parents, my sister Gema and my grandparents for their unconditional support which has been invaluable.
- And finally, I would like to give a special acknowledgment to my wife Diana. Without her help I have not been able to finish this dissertation. She has helped me to see the live with other eyes. She always says me that life is wonderful and she is right. Thanks for your daily smile and for giving those hugs that I always need.

Chapter 1

INTRODUCTION

1.1 Wireless Sensor Networks

With the last advances in wireless communications and digital electronics, the design and development of low-cost, low-power, multifunctional sensor nodes have become possible. Nowadays, they are small, smart and they can communicate in short distances with other sensors, needing few energy. All capabilities of these sensor nodes, which include sensing, data processing, and communicating, enable to make wireless sensor networks (WSNs) based on the collaborative effort of a large number of sensor nodes.

Wireless ad hoc networks (WAHN) are simple networks in which a coordinator is not needed and the numbers of nodes and network topology are not predetermined. A WSN is a type of WAHN composed of nodes with sensing capability. There are several differences between WSN and WAHN [1]. WSNs usually have a larger number of nodes and are deployed in close proximity to the phenomena under study; the nodes mainly use a broadcast communication paradigm and the network topology can change constantly due, for example, to the fact that the nodes are prone to fail (they have limited power, computational capabilities and memory). Mobile wireless sensor networks (MWSNs) are WSNs with mobile sensors which are randomly deployed in an interesting area for sensing some phenomena. These mobile sensors collaborate with each other to form a sensor network with the capability of reporting sensed phenomena to a data collection point called sink or base station.

On the other hand, a mobile ad hoc network (MANET) [2] is a self-configuring network of mobile nodes connected by wireless technology. This type of network has an arbitrary topology. The network's wireless topology may change rapidly and unpredictably. Independently of the medium access method used [3], in recent years have many routing protocols been developed for these networks [4]. The nodes' mobility, the lack of stability of the topology, the lack of a pre-established organization and performing of the wireless communications are the reasons for not using the routing protocols developed for fixed networks. Therefore, MANETs and WSNs are networks with self-configuring nodes connected by wireless links, where resources are scarce, and traditional protocols and network algorithms are inadequate.

However, we must be very careful before applying algorithms, protocols and techniques for WSNs, if they were originally developed for MANETs. Although both types of networks have many similarities, there are differences and these reasons WNSs are considered a different research field [5]. Some of these differences are derived from uses of these networks: MANETs are usually intrinsic to humans, network devices are designed to be handled by people (i.e. laptops, tablets, smartphones, etc.); in contrast to this, the nodes of WSNs do not communicate directly with people, they interact with environment. The nodes in a WSN are often located in a strategic place. They sense several phenomenon, which are happened near to their. Some people think that this type of networks could be considered as “macroscope”. As a consequence of this, the number of nodes and the density of nodes can vary a lot, but it is always greater than in MANETs. So this feature involves another important issue in WSNs, the scalability.

If a network is implemented in an outside environment like on top a volcano, on sea, or on another place where the nodes could be damaged in an easy way, they could induce several errors and it brings out a big fail in the network. That means a WSN topology can change dynamically not due to node’s mobility in an ad hoc network, but by the failure of some nodes. In these cases the reconfiguration processes will have to be implemented to satisfy the fault tolerance feature. Besides that, a topology may also change due to sleep-wake cycles observed in some protocols for sensor networks. These kinds of protocols are based on these cycles in order to save energy, which are a major concern and an essential requirement in the design process because WSNs have few resources. This last feature is another differentiator between mobile networks and WSNs. The nodes of WSNs are abandoned for long periods of time (i.e. months or years) and they must be able to work with batteries. Wireless communications coverage is around 50 meters and the date rate are usually low (some kbps). The nodes have little memory capacity and its CPU works with some megahertzs, as we can see in the next chapter.

It is important to know that the goal of the wireless sensor networks is not only send information from a node to another one, a sensor node can offer some replies too. A node must reply to questions like: What are the regions of the network where the temperature is above the specified threshold? What is the path followed by a flock? This feature is not necessary in other networks. In fact, in some applications the ID (identifier) of a sensor node is irrelevant, but its position could be more important. In general, the most of communication’s algorithm is related to final application. In some cases, the ID of each node is important because we want to know who sends these data, but in other situations can be more important the value of data.

Depending on the type of the information exchanged by the nodes and on the frequency by which they do it, the routing protocols in ad hoc networks are divided into three types: proactive, reactive and hybrid.

- The proactive protocols update the routing tables of all the nodes periodically, even though no information is being exchanged. When a topology change occurs, the routing table is updated and the routing protocol sends the best route to forward the information. A periodical control protocol message exchange allows this, but consumes bandwidth and energy.

- The reactive protocols only maintain routing routes in their tables when a node has to communicate with another node in the network. With these protocols, when a communication starts, as the right route is unknown, a route discovering message is sent. When the response is received, the route is included in the routing tables and the communication is established. The main disadvantage of these protocols is the latency at the beginning of the communications (route discovery time) but they improve the consumption of network and energy resources.
- Finally, hybrid protocols are a combination of the above two types, taking their advantages. These protocols divide ad hoc networks into different zones; consequently near nodes use proactive routing while far nodes use reactive routing.

The aforementioned networks and protocols do not have a predetermined topology, so they could be applied over different types of architectures such as regular, grids, cluster-based, group-based and so on.

A key problem in the planning of any kind of network is to design the communication topology. It means deciding how the nodes are connected as well as how their messages are exchanged. Topologies can be characterized by several parameters such as the number of nodes in the network, the number of links or connections (hereafter both terms will be used without distinction in this dissertation) in the network and their bandwidth, the degree of the nodes and the diameter of the topology. On the other hand, communication topology design needs to address several conflicting requirements like, on the one hand, minimizing the overall network diameter, minimizing the convergence time, the infrastructure cost (total number of links), the book-keeping costs (the number of links maintained by each node) and the management cost, and, on the other hand, maximizing load distribution, reliability, efficiency, fault tolerance, the performance of the system, the scalability, and so on.

Usually, optimizing on any requirements would be at the cost of others. Designing the optimal topology for a given set of constraints is a difficult problem. Over the years, topology design has received significant interest in many areas. In order to provide real-time infrastructures, reliable, available and efficient networks and QoS-aware distribution services, a topology-aware network is necessary [6].

While the physical topology defines how the nodes on a network are physically connected and the physical layout of the devices on the network, the logical topology defines how the nodes on the network communicate (i.e., the way the data passes through the network, with no regard for the physical interconnection between the devices). However, if the logical network is constructed randomly, nearby hosts in the logical network may be far away in the physical network. This may waste too many network resources, and hence degrade data delivery performance significantly. So, for these reasons we propose a group-based logical topology for WSNs, which improves the network’s behavior.

The main motivation of this thesis is to create a group-based architecture to improve the efficiency of WSNs, regardless of the protocols that are being implemented in the lower layers. This architecture is located over the transport layer, so it does not depends on any routing protocol. Nowadays, as we can see in Chapter 2, every protocol based on groups or clusters uses its own routing protocol. We use groups instead of cluster because cluster-based networks are a subset of the group-based networks. Every cluster could be considered as a group. But a group-based network is capable of having any type of topology inside the group, not only clusters. It is more general.

Collaboration between groups of humans improves the society, problem solving, ability to work and communication. So, if this feature was extrapolated to networks and wireless sensor networks, what could happen? This was other motivation to include this feature to our group-based architecture. The main idea about this motivation is every group collaborates with its neighbors in order to give some information about its group. This information is used by the neighbors groups to manage its energy, alarms, etc.

Finally, other important motivation is to develop a system, which uses our group-based architecture to communicate the sensor nodes. The creation of an end system is always a motivation, because the goal is to create something that will give benefits to society. From my point of view, the motivation to create a product is always superior to the motivation for developing a theory. But, most of the time without a good theory should not have a final product. In our case, this system is to make a group-based WSN in order to manage a marine fish farm and controlling sustainability of the seabed.

1.2 Research Aims

In this subsection, we present the following objectives and the steps involved in this procedure to achieve our main idea, which is to divide our network into several groups to give better performance to network regardless of routing protocol. In order to develop this idea and improve it, we must carry out the following tasks:

1. Analyzing and checking the behavior of wireless sensor networks when they are using group-based topologies.
    a. Studying several types of grouping nodes in networks.
    b. Simulating several scenarios where we can analyze the network performance when they are using group-based or standard topologies.
    c. Showing conclusions about what types of ad hoc network protocols are better for these group-based topologies.
2. Developing, programming and simulating our group-based architecture and protocol for wireless sensor networks.
    a. Analyzing our group-based topology using mathematical theory.

   b. Designing our proposal using flow charts, state machines, etc. In this development will be analyzed several topics like: group's creation, neighbor's selection, group's balance, etc.
   c. Programming and developing every topic developed previously in order to make our simulations.
   d. Simulating our proposed architecture and protocol in order to check the performance of network.
3. Adding new features in our group-based architecture for WSNs.
   a. Including collaborative feature in order to save more energy in our group-based architecture.
   b. Developing and test the intergroup collaboration algorithm.
   c. Adding collaborative data fusion in order to improve the network's behavior.
   d. Including security and time synchronization in our proposal.
   e. Developing and testing our security protocol and showing as our synchronization algorithm works.
4. Adapting our system in order to be used in a marine real environment.
   a. Looking for the needs required in this type of environment.
   b. Proposing a group-based application in order to satisfy the requirements of final application.
   c. Simulating the network's behavior when it is being used in this real environment.

1.3 Previous Works

The idea to create logical groups in networks is not entirely new. An example in the wired networks are VLANs (Virtual Local Area Network, [7]), whose main function is to create independent logical networks within the same physical network. In this case, the grouping is performed in the data link layer, but this grouping can also take place at higher layers.

The starting point of this dissertation is the conclusion drawn in the thesis of my supervisor Jaime Lloret [8]. This thesis was proposed a hierarchical system of autonomous networking for sharing files between P2P networks. This proposal shows the improved performance relative to a standard architecture. System's adaptation for wireless sensor networks was not trivial and it does not make sense due to the characteristics of WSNs. For instance, in P2P networks there are not problems with distance between nodes, but in WSN this is a problem. This feature makes more difficult to join the group headers to create a hierarchical network. Therefore, we develop a planar group-based architecture in order to improve the efficiency in the communications of WSNs.

Another idea born from that thesis was presented in reference [9]. This paper shows the development of an architecture that creates clusters and establishes connections between sensors of the same type by building different sensor networks. Cluster heads (CH) manage the network since they have connections with other cluster heads and these connections allow connecting cluster members from different clusters. Cluster members of the same type form a specialized network. Although there are several proposals of cluster-based systems in existence, the novelty of our proposal is that it could be used to build different networks with different routing protocols, while other cluster-based networks can run just one routing protocol and can build only one type of network. One of the main goals is that if all cluster heads switch off at the same time, the system is able to continue working, although there will not be new connections between clusters through CHs.

While we have done this thesis has come out another work [10]. In this thesis an innovative solution is proposed, aimed primarily at reducing energy consumption. The new architecture called EDETA (Energy-efficient aDaptative hiErarchical and robusT Architecture) is also scalable, suitable for both homogeneous WSNs and heterogeneous ones (increasingly employed), able to self-configure, which supports multiple sinks in a transparent manner, and provides features such as fault tolerance and bounded time responses, without degradation of the network performance. The proposed architecture is based on a two-level hierarchy. The lower level is based on clusters, governed by an internal protocol called Intra-Cluster Communication, while the upper level consists of a dynamic tree node of cluster leaders, the so called Inter-Cluster Routing protocol. This proposal is completely different to our proposal because as we have seen in the previous subsection our group-based network is based on a planar architecture.

There are several works related to this thesis, but all of them have been used to research new concepts about this type of networks. For this reason, these works will be referenced in their corresponding chapters in order to introduce ideas or solution in each chapter.

1.4 Main Contributions

In this section we are going to show a summary about every contribution performed on behalf this doctoral thesis. These contributions will show following a temporal order, because in this way we can see the evolution of this thesis.

My first contact with group-based architectures was with the paper [11].This paper proposes grid architecture based on groups. The architecture organizes logical connections between nodes from different groups of nodes allowing sharing resources, data or computing time between groups. Connections are used to find and share available resources from other groups and they are established based on node's available capacity. Using these ideas, we proposed our first group-based architecture over WSNs in [12]. Where, we propose to divide the network into several groups of sensors. When a sensor send data to other groups, the data has to arrive just to one sensor from each group, then they propagate it to the rest of sensors in their groups. This first step was improved with the paper [13]. In this work, we show the development of a suitable neighbor selection strategy for group-based wireless sensor networks that is based on a capacity parameter defined by us and the new neighbor distance.

At the same time, while we were proposing a new group-based architecture for WSNs. We had to prove that group-based topologies improved regular topologies. For this reason we did a thorough study to show the performance of group-based networks. This performance study was published in [14], [15], [16] and [17]. After showing that group-based topologies had a better performance than regular topologies we propose a group-based architecture for WSNs with more details. This work was published in [18], [19] and [20]. In these papers the network architecture is described with its mathematical description and the messages that are needed to proper operation. It is also simulated how much time is needed to propagate information between groups. A comparison with another group-based architecture is shown.

After making this group-based architecture we thought about adding new features to our proposal. The first one was to include collaboration between groups. This first idea was presented in [21]. In this work, we propose a monitoring group-based sensor network which uses the cooperation between groups. When a group detects an event, it warns the alert, jointly with the parameters measured, to its neighboring groups. Cooperation with other groups could change the direction of the alert propagation and the level of the alert. According this cooperation, the sensor network will be efficient and the sensors will have a longer lifetime. Following this idea and introducing new knowledge we went improving this feature. This evolution about collaboration in group-based WSNs was published in [22], [23], [24] and [25].

The next step was to include security in our group-based wireless sensor network. This feature was published in [26]. First, in this paper we explain the methods of key creation and how they are used in our proposal. Then, the key management method, which is used to determine the tasks performed by the nodes to maintain the keys updated, is detailed. The process of secure communication between groups is also explained. Finally, energy consumption measurements for each protocol operation are shown. Reviewing the most important features in WSNs, we saw that synchronization is a fundamental element because it controls time-stamping in measurements, signal processing in the network, localization, cooperative communication, media access, sleep scheduling and coordinated actions. For these reasons we propose a secure intragroup time synchronization technique to improve the security and performance of group-based wireless sensor networks [27].

Then, we think that it was necessary to find a real environment where we can set up this group-based architecture and protocol. In papers [28], [29] and [30] we propose a system based on sensor arrays for monitoring and managing a marine farm, where there are usually different cages.

Finally, during these years doing this thesis we have had to know how wireless sensor networks work in detail. Due to that, we have had some surveys and reviews about WSNs and underwater WSNs. From the performed studies we have published the following papers [31], [32], [33], [34], [35] and [36]. They are related to energy studies in WSNs, WSN's deployment surveys and general features of this type of networks and finally, a survey about wireless sensor networks and communications in unwater environments.

1.5 Structure of the Thesis

After entering the main issues that have motivated this thesis and the main objectives pursued, the rest of the memory is organized as follows.

In Chapter 2, we show a detailed state of the art of several issues (e.g. grouping, collaboration, security, synchronization, etc.) in group-based wireless sensor networks. Firstly, we make an introduction about topologies in WSNs, particularly we define the concept of a group as a small number of interdependent nodes with complementary operations that interact in order to share resources or computation time, or to acquire content or data and produce joint results. Also, we show the main differences between these networks and other networks such as cluster networks, LANMAR, etc., which might be a subset of group-based networks. Moreover, we have introduced several related works about collaboration in networks, and how this feature improves the efficiency of the network. Security in wireless sensor networks, we have focused our attention in some works where their authors apply security in group topologies. Then, we have explained several related work about synchronization topic, which is very important in this kind of networks. Finally we have shown some applications where group-based topologies could be applied.

Then, in Chapter 3, we show the main differences between these networks and other networks such as cluster networks, which might be a subset of group-based networks. In this chapter we show that the group topologies provide improvements over standard topologies. These improvements have been studied by simulations of WSNs. Several networking data have been analyzed on many network topologies. With these simulations we can see that the group-based topologies improve network performance in general terms but they introduce more management traffic. Finally, we have presented various application environments where they could be properly implemented using group-based topologies.

Chapter 4 presents our group-based proposal. This proposal is defined by two elements. Firstly, we make an introduction about topologies in WSNs, particularly we focus our attention to see the differences between the existing architectures, such as cluster networks, LANMAR, etc. and our proposal. When we know that our proposal is different than the previous works we start defining several issues about our group-based architecture, i.e. types of nodes and their features, connections between nodes, etc. Then, inside this architecture we analyze how a node selects a neighbor and how the number of nodes varies according to network topology. Furthermore, we will show in detail the messages needed to carry out this group-based architecture. Secondly, we will describe the operations of our group-based protocol in order to manage our groups in a WSN. We describe how to create and manage a group, what do when a node fails or leaves its group, etc. Finally, we will present some connection time simulations to show the time needed to send information from a group to another and we will show our conclusions about this chapter.

The implementation of our group-based protocol is explained in Chapter 5. This chapter starts with a small introduction about programming model in WSNs. At the end of this first section we take into account that these programming movements are changing and nowadays a lot of companies are using C, as a programing language to implement things in WSNs. The next step is to present the tools and the programming language used. We explain OMNet++ simulator, which are its features, its modeling concepts and how it works using C++. In section 5.3 are explained every class used in our implementation. We have used UML diagrams to show the classes' relation. This implementation is checked in several simulations and we present them in section 5.4. Finally, our conclusions about this chapter are detailed in the last section of this chapter.

An improvement about group-based topology explained in the previous chapter it is shown in Chapter 6. This chapter presents the concept of collaboration between groups in a group-based network. This feature gives a lot of advantages, which are detailed in this chapter. It is organized as follows. Firstly, we introduce some related works about collaborative term related to networking environments. Then, we describe differences between regular architectures and collaborative group-based architectures. A collaborative group-based network is like a group-based network, where groups collaborate between them in order to improve the behavior of the network, i.e. power saving, sending few information, etc. In the next section inside the same chapter, we analyze the energy used in collaborative group-based WSNs. With this analytical model we find out that a WSN needs less energy when it uses the collaborative feature. After that, we analyze if this feature induce any improvement when we are talking about communications (i.e. number of messages needed). In order to test these analytical models we present some simulations, which prove improvements made by such architectures. Besides, in this chapter, it is explained the algorithm needed to keep the intragroup communication in these collaborative group-based WSNs. Lastly, we show an example of collaborative group-based WSNs, where the communication between collaborative groups and data fusion is essential for their proper operation.

Chapter 7 shows us how we could add security and synchronization in group-based wireless sensor networks. Initially, in this chapter we introduce the concept of security in WSNs and what encryption algorithms are the best to use in these networks. Then, it is explained the method used to maintain updated key in every sensor node. Also, it is shown exchanged messages to manage the keys when a sensor node changes to another group when we are adding new features in WSNs, which involve running batteries down. For this reason, we show some measurements to check that this proposed system does not worse a lot the energy saving. In addition to this, we introduce a secure time synchronization system to preserve the synchronization between the sensor nodes inside the same group. In this section we explain why a WSN needs time synchronization techniques, what the techniques are used currently and the most frequent error sources. Our secure time synchronization algorithm is proposed and it is simulated in order to know its performance.

After showing our groups based architecture for WSNs, introducing enhancing features such as collaboration between groups, data fusion, synchronization and security. In Chapter 8 we show a possible real application where our group-based architecture could be applied. In this chapter we propose a system based on sensor arrays for monitoring and managing a marine farm, where there are usually different cages. First, we show what the main problems are in the existing marine farm. Knowing these problems we propose to use an array of sensors for monitoring all elements, which may cause some problems listed above. Finally, the sensor array system and distributed sensor data fusion system are explained for proper feeding in a marine farm.

Finally, in Chapter 9 we show a summary of conclusions viewed in each previous chapter, because every chapter has its own conclusions. Moreover, in this chapter we describe some future works related to our group-based architecture and protocol.

Chapter 2

STATE OF ART IN WSNS WITH GROUP TOPOLOGIES

2.1 Introduction

Latest advances in sensor technology are leading to the development of distributed mechanisms and small devices with both low cost and low energy consumption. In addition, these devices are capable of processing information locally and communicating wirelessly with other elements. These devices are called sensor nodes or motes.

In some cases, amount of sensors are necessary to sense an environment or take measurements from the surroundings. While they are sensing, they have also to communicate between them and/or with a central server. On the other hand, a monitored environment doesn't have infrastructure to supply energy for communication. So, motes must work with small batteries and use wireless channels. Another feature of the sensor networks is their capacity of distributed processing. It is necessary because, the communication is the process that consumes more energy. A distributed system means that some sensors need to communicate through long distances. So, it is a good idea to process locally data as much as possible in order to minimize the bit rate.

Wireless Sensors Networks (WSNs) are formed dynamically because the connectivity between nodes depends on their position and their position variation over the time (if they are mobile). This kind of networks is characterized as being easy to be deployed and self-configuring. A sensor node is composed by a transmitter, a receiver, and it offers services of routing between nodes without direct vision, as well as records data from other sensors.

The following are some of the main features of WSNs:

- Dynamic topologies: In a wireless sensor network, the topology is always changing because nodes can fail or new nodes can join the network. These changes affect the communication between sensors.
- Variability channel: The radio channel is highly variable. There are several phenomena, such as the attenuation, fast fainting, slow fainting and interference that can cause data errors.

- Ad hoc networks: Generally, sensor networks do not have a wired network infrastructure. All motes are transceivers and routers simultaneously. However, the concept of sink node is important; this node collects the information and sends it to a central computer capable of processing these data.
- Failure tolerance: A sensor node should be able to continue operating despite of the existence of errors in the system.
- Multi-hop or broadcast communications: This type of networks use any routing protocol to enable communications multi-hop, although it is also very common the use of messages sent in broadcast.
- Power saving: It is one of the most important features in these networks. Currently, the motes have limited energy. A sensor node should have an ultra-low consumption processor and transceiver radio. It is one of the most restrictive features.
- Limited hardware: In order to get an adjusted consumption, the hardware should be simple; this brings a limited process capacity.
- Production costs: Sensor networks are formed by high number of nodes. Motes must be economic to create a reliable network.

Since sensors can collect data from environment, a sensor network has many application areas, such as habitat monitoring, fire detection, motion tracking, reservoir water controlling, or intruders controlling. In order to control, monitor, tracking or detect, it is necessary a large quantity of sensor nodes which detect the monitored event (light, pressure, sound, heat, humidity, electro-magnetic field, proximity, location, etc.), and transmit it to a base station, where last action will be made. Sensor networks have become very useful for our lives and they have penetrated in domains such as health, home care, environment monitoring, etc. [37].

In WSNs, topology refers to not only the locations of the nodes but also their activity states, i.e., which nodes are active for communication, as well as the links between them. Hence, several techniques exist for topology management. The four main topology management techniques are shown in **Figure 2-1**. These techniques can be used individually as well as in conjunction with each other since different properties of the topology are controlled by each one.

• Deployment: The first phase of topology management is network deployment as shown in **Figure 2-1 a)**. Deployment techniques determine the locations of the nodes in the network, which plays an important role in the coverage as well as the connectivity of the network. Accordingly, the maximum area that can be covered by the on-board sensors can be maintained. Similarly, deployment affects the links that can potentially be formed given the limitations of the wireless channel.

• Power control: The topology is partly defined by the links between the nodes. In a wireless network, these links depend on the capabilities of the transceiver. Power control solutions control the transmit power of the transceivers to maintain the communication range of a node as shown in **Figure 2-1 b)**. Accordingly, the number of neighbors of a node or the hop length in a multi-hop path can be determined for energy efficiency, reliability, or latency goals.

• Activity control: In a WSN, the transceiver of the sensor node can be turned off during certain periods to increase energy efficiency. As a result, certain nodes become disconnected from the network during their sleep states. Hence, the network topology can also be maintained by managing the activity of the sensor nodes as shown in **Figure 2-1 c)**. Accordingly, redundant nodes can be turned off while still maintaining the connectivity and capacity of the network.

• Clustering: In addition to flat topology-based techniques, the network can also be partitioned into clusters as shown in **Figure 2-1 d)** to improve scalability and energy efficiency. In this case, a group of nodes are placed into clusters, which are controlled through cluster heads (CHs).

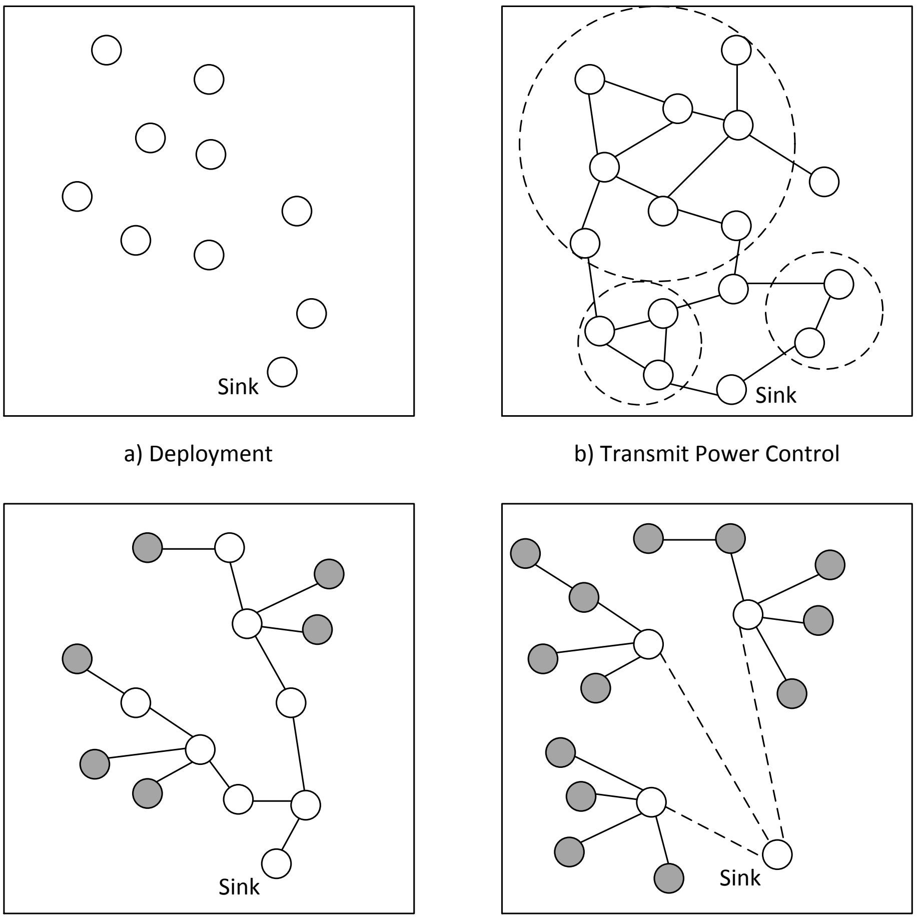


Figure 2-1: Topology management techniques.

In this chapter we review different group-based architectures available today. First, we are going to analyze the topologies in a data network, and then we focus on the topologies on groups based WSN. Later, we will analyze several aspects of group-based topologies.

2.2 Group-based Topologies

Network topology defines how the nodes on a network are physically or logically connected (i.e., the physical layout of the devices on the network). Three types of network topologies can be distinguished:

- Centralized Networks. In these topologies there could be no direct connection between nodes, and all nodes' messages could be mediated by a mediator, generally known as a central node. This single node acts as a gateway for all the nodes. These topologies have been used for many types of networks [38].
- Decentralized Networks. Each node is able to connect directly with all other nodes, and messages are sent without intermediation via a central node. All nodes have the same responsibility and functionality in the network. No element in the network is essential for the system operation. A node in a decentralized topology can play three roles: server, client and router. Many types of networks have decentralized topologies, such as pure peer-to-peer networks, ad-hoc and sensor networks, and grids. Many searching algorithms for decentralized networks have been designed [39], all of which perform three basic actions: searching of active nodes, querying for resources or services, and content transferring.
- Partially Centralized Networks (also known as hybrid networks, layered networks or multi-tier networks). In these networks, there are some nodes with higher roles which form the backbone of the network and are needed to run the system. Nodes with the lower role are called leaf nodes and will be placed in the lower logical layer, while nodes with higher roles could be supernodes and will be placed in the higher logical layers. Every supernode or leaf node can have connections with either leaf nodes or supernodes. There is a hierarchy where higher layer nodes organize, control or gather data from lower layer nodes. Higher layer nodes are used to forward messages from lower layer nodes. Layered networks have been used for different types of networks such as wireless networks [40].

Let us suppose we need to divide the network into groups or areas due to the physical implementation of the WAHSN or due to scalability purposes. It does not matter which kind of routing protocol is being used inside each group. All architectures shown above fail to solve that problem efficiently, because in the case of centralized architectures, the server will have many wireless connections at the same time, so it will need many resources. There is also a central point of failure and a bottleneck. On the other hand, in the case of fully distributed architectures, it is very difficult to control the system and it needs a long time to process tasks (due to the time needed to reach far nodes), decreasing the performance of the system.

We propose to divide the whole WAHN or Wireless Sensors and Actor Networks (WSAN) into several groups and when a node receives data for its group, it will propagate the data to the rest of the nodes in its group.

A group is defined as a small number of interdependent nodes with complementary operations that interact in order to share resources or computation time, or to acquire content or data and produce joint results. In a wireless group-based architecture, a group consists of a set of nodes that are close to each other (in terms of geographical location, coverage area or round trip time) and neighboring groups could be connected if a node of a group is close to a node of another group. The main goal in a wireless group-based topology is the network protocol and the group management, that is, the design of an efficient algorithm and a capable protocol is needed to find the nearest (or the best) group to join in when a new node appears in the network. The performance of the network largely depends on the efficiency of the nearby group locating process and on the interaction between neighbor groups.

We have to distinguish between groupware architecture and a group-based architecture. In groupware architecture all nodes collaborate towards the correct operation and the success of the network purpose, while in a group-based architecture the whole network is broken down into groups and each group can perform different operations or can have different routing protocols.

Some important issues must be taken into account in a wireless group-based architecture regardless of the protocol inside the group:

1. How to build neighboring groups.
2. A protocol to exchange messages between neighboring groups.

We can distinguish two types of group-based topologies: Planar group-based topologies and layered group-based topologies. In planar group-based topologies all nodes perform the same roles and there is only one layer. However, in some works there is a directory server or a rendezvous point (RP) for content distribution coordination. Nodes from layered group-based topologies could have several roles (2 roles at least). Depending on which type of role they are running, they will belong to a specific layer. All nodes in the same layer will have the same role. There will be connections between nodes from the same layer and from different layers, but these layers must be adjacent. We have included hierarchical architectures in this group, because the hierarchies could be considered as layers. There are several differences between both group-based topologies. While layered group-based topologies grow in a structured form, organized by upper layers, planar group-based topologies grow in an unstructured form, without any organization. On the one hand, in layered group-based topologies any node can know exactly where each group is and how to reach it; on the other hand, planar group-based topologies, due to the groups join the network as they appear, every time there is a connection between nodes from different groups, the message should travel through many unknown groups in the path. Delays between groups in layered group-based topologies could be lower because connections between groups can be established taking this parameter into account. In planar group-based topologies, connections between groups are established by the group's position, their geographical situation or their appearance in the network. Layered networks involve some complexity because nodes could have several types of roles and fault tolerance must be designed for each layer. Planar networks are simpler because all nodes have the same role. In order to be more scalable, layered group-based topologies must add more layers to its logical topology, while planar group-based topologies could grow without any limitation, just the number of hops of the message.

Group-based networks provide some benefits for the whole network, such as the following:

- Spread the work efficiently to the network in groups giving more flexibly, and lower delays.
- Content availability will increase because it could be replicated in other groups.
- Anyone could search and download data from every group using only one service.
- Fault tolerance. Other groups could carry out tasks from a failed one.
- Scalability. A new node can join any group and a new group could be added easily.
- Network measurements could be taken from any group.

There are some works in the literature where nodes are divided into groups and connections are established between nodes from different groups, but all of them have been developed to solve specific issues, but none of them for WAHSN networks as we can as follows.

The Rhubarb system [41] organizes nodes in a virtual network, allowing connections across firewalls/NAT (Network Address Translation), and efficient broadcasting. Nodes can be active, if they establish connections, or passive, if they don’t do it. The Rhubarb system has only one coordinator per group and coordinators could be grouped hierarchically. It uses a proxy coordinator, an active node outside the network, and all nodes inside the network make a permanent TCP connection with the proxy coordinator, which, if broken, can be renewed by the firewall or NAT. When a node from outside the network wishes to communicate with an inner node, it sends a connection request to the proxy coordinator, which forwards the request to the inner node.

A Peer-to-Peer Based Multimedia Distribution Service was presented in [42]. They propose a topology-aware overlay in which nearby hosts or peers self-organize into application groups. End hosts within the same group have similar network conditions and can easily collaborate with each other to achieve Quality of Service (QoS) awareness. When a node wants to communicate with a node from another group, the information is routed through several groups until it reaches its destination.

There are some hierarchical architectures where nodes are structured hierarchically and parts of the tree form groups, such as the ones in references

[43] and [44]. In some cases, some nodes have connections with nodes from other groups although they are in different layers of the tree, but in all cases, the information has to be routed through the hierarchy. Hierarchical protocols aim at clustering the nodes so that cluster heads can do some aggregation and reduction of data in order to save energy. Some of the hierarchical protocols proposed in the literature designate a cluster-head different from the normal sensors. These cluster-heads could be more powerful than the sensor nodes in terms of energy, radio coverage, bandwidth and memory.

There are many cluster based hierarchical architectures [45]. In cluster based architectures the mobile nodes are divided into virtual groups. Each cluster has adjacencies with other clusters. All the clusters have the same rules. A cluster can be made up of a Cluster Head node, Cluster Gateways and Cluster Members [46]. The Cluster Head node is the parent node of the cluster, which manages and checks the status of the links in the cluster, and routes the information to the right clusters. The rest of the nodes in a cluster are all leaf nodes. In this kind of network, the Cluster Head nodes have a total control over the cluster and the size of the cluster is usually about 1 or 2 hops from the Cluster Head node. The cluster gateways have links to other clusters and route the information to those clusters. On the other hand, a cluster member is a node without any inter-cluster links. Finally, we want to emphasize that the cluster-based networks are a subset of the group-based networks, because every cluster could be considered as a group. But a group-based network is capable of having any type of topology inside the group, not only clusters. However, both types of networks have been created to solve the scalability problems of the WAHSN.

An example is LEACH [47]. It is one of the most popular hierarchical routing algorithms for sensor networks. The idea is to form clusters of the sensor nodes based on the received signal strength and use local cluster heads as routers to the sink. This will save energy since the transmissions will only be done by such cluster heads rather than all sensor nodes. Optimal number of cluster heads is estimated to be 5% of the total number of nodes. All the data processing such as data fusion and aggregation are local to the cluster. Cluster heads change randomly over time in order to balance the energy dissipation of nodes. An improvement of the LEACH protocol is PEGASIS [48]. PEGASIS forms chains from sensor nodes so that each node transmits and receives from a neighbor and only one node is selected from that chain to transmit to the base station (sink). Gathered data moves from node to node, aggregated and eventually sent to the base station. PEGASIS has been shown to outperform LEACH by about 100–300% for different network sizes and topologies. However, PEGASIS introduces excessive delay for distant node on the chain. In addition the single leader can become a bottleneck.

TEEN [49] is a hierarchical protocol designed to be responsive to sudden changes in the sensed attributes such as temperature. It pursues a hierarchical approach along with the use of a data-centric mechanism. The sensor network architecture is based on a hierarchical grouping where closer nodes form clusters and this process goes on the second level until base station (sink) is reached. After the clusters are formed, the cluster head broadcasts two thresholds to the nodes. These are hard and soft thresholds for sensed attributes. As a consequence, soft threshold will further reduce the number of transmissions if there is little or no change in the value of sensed attribute. One can adjust both hard and soft threshold values in order to control the number of packet transmissions. However, TEEN is not good for applications where periodic reports are needed since the user may not get any data at all if the thresholds are not reached. APTEEN [50] is an extension to TEEN and aims at both capturing periodic data collections and reacting to time critical events.

GAF [51] is an energy-aware location-based routing algorithm designed primarily for mobile ad hoc networks, but may be applicable to sensor networks as well. Although GAF is a location-based protocol, it may also be considered as a hierarchical protocol, where the clusters are based on geographic location. For each particular grid area, a representative node acts as the leader to transmit the data to other nodes. The leader node however, does not do any aggregation or fusion as in the case of other hierarchical protocols.

We can also find in the literature a routing protocol based on zones. It is the Zone Routing Protocol (ZRP) [52]. Each node proactively maintains routing information for a local neighborhood (routing zone), while reactively acquiring routes to destinations beyond the routing zone. ZRP and our proposal have several common features, e.g. they could be applied over any type of routing protocol, they scale well and the information is sent to end nodes in order to reach destinations outside their zones. The main difference between them is that in ZRP each node maintains a zone and the nodes in that zone have different nodes in their zone while in our proposal all the nodes that form a group have the same nodes in their group.

On the other hand, we will not consider other works of groups systems such as the following. The community based mobility model for ad hoc network research presented in [53]. Although the network is organized in groups, their nodes can move from one host to another, there is not any connection between end nodes from different groups. The landmark hierarchy presented in [54] because although there is a node with higher role which has connections with nodes from other groups, its leaf nodes do not. Another example similar to the last one is the BGP routing protocol architecture [55]. Finally, we will not consider moving groups such as Landmark Routing Protocol (LANMAR [56]), where the set of nodes move as a group, so the group can enlarge or diminish with the motion of the members.

We want to highlight that there is no protocol such as the one that is presented in Chapter 4 of this dissertation. It is based on the formation of groups of sensors instead of clusters of sensors. Moreover, there is cooperation between nodes in order to minimize the energy consumed in the network. A group-based network is capable of having any type of topology inside any group, not only a cluster. Furthermore, in a group-based network, each group could be using a different type of routing protocol. A very nice explanation about cluster-based networks is presented in [57]. Group-based topologies are also explained in [58]. A summary of the differences between cluster-based and group-based can be seen in **Table 2-1**.

Table 2-1: Differences between cluster-based and group-baseed networks.

	Cluster-based network	***Group-based network***
Hops between the main node and farthest node	1 or 2	Minimum 1
Topology inside each cluster/group	All the same	Different in each group
Can the main node be a gateway?	Yes	No
Functions of the main node	Total control over its cluster	Create and manage the boundary of each group
Type of architecture	One-hop Planar Hierarchical	Planar
Allows different routing protocols	No	Yes
Cluster/Group size determinate by	Hops	Several metrics

2.3 Collaboration in Group-based Topologies

No one of the papers found in the related literature analyze the power consumption and the communications using the group-based strategy (except the in a paper from the same authors [12], where we introduce these ideas). On one hand, some works show that organizing the sensors into groups provide greater benefits than doing otherwise. On the other hand, some works are focused on network architectures to decrease the energy consumption, but without forgetting the communications efficiency.

There are also several published works related with the analysis of energy consumption and energy saving mechanisms in WLANs. These types of analysis may help us know some power saving issues when they are used in WSNs. The paper [59] describes several techniques to reduce the dynamic energy consumption in WLAN IEEE 802.11n standard systems in order to increase the battery life in mobile systems. They propose to reduce the devices consumption from their initial design by building smaller devices because they consume less energy.

V. Raghunathan et al. describe in [60] some considerations about the architecture and protocols to be considered in order to make an energy-efficient design of a sensor node in a WSN. The paper shows an analysis of the energy consumption characteristics of a node, using multihop architectures. Moreover, they identify the key factors that can affect the life of the global system. They show the sensor energy consumption, the power-aware computing, the energy-aware software, the power management of wireless communication, the energy-aware packet forwarding and traffic distribution, among others. This work is important to know what the most important factors are, in order to take them into account when a system is developed.

In paper [61], the authors discuss the required key technologies for low-energy distributed microsensors and present a power aware Application

Programming Interface (API) to calibrate the energy efficiency of various parts of the application. They took care of the power aware computation/communication component technology, low-energy signaling and networking, system partitioning considering computation and communication trade-offs, and a power aware software infrastructure. This work also shows some analytical models of the device behavior and the results of applying different parameters, such as the number of sensors, type of protocol used to communicate the sensors, the system power consumption by controlling when they should enter a sleep mode and applying dynamic voltage scaling. These models will help us to develop our model.

Reference [62] shows the relationship between the power usage and the number of neighbors in a WSN. They state that many of the topologies proposed for wired networks cannot be used for wireless networks because wireless networks depend on the physical neighborhood and the transmission power while in a wired network depends on the physical connections among nodes. A. Salhieh et al. analyze various 2D and 3D structures with different numbers of nodes. They measured the network performance for different network topologies and the node's power dissipation in order to determine what the best topology is for a WSN. But, they assume that they control the placement of these sensors and the sensor locations are fixed respect to each other. Moreover, the authors do not consider the effects of communication with a base station. They conclude that the best power efficiency in 2D topologies is achieved with four neighbors, and although 3D topology is better, it may not be feasible for some applications.

The energy consumption must not deteriorate the communications quality. We have found some works that attempt to improve the efficiency of the communications in WSNs, but trying to reduce energy consumption. In [63], the authors analyze the broadcast routing according to the energy cost. In this case, the energy cost is defined as the sum of each individual node energy cost that transmits broadcast messages. So, in order to solve this problem, the authors try to minimize the number of broadcast messages. They use three centralized heuristic methods to minimize it. According to their simulations, we can see that there are some algorithms better than others. But the main problem of these solutions is that all of them are based on centralized solutions and nowadays the architectures are evolving to decentralized solutions.

In [64], the authors show that the cluster-based sensor networks have a better energetic behavior than regular sensor networks. They present a routing protocol for managing the sensor network with the main objective of extending the life of the sensors in a particular cluster. Their proposal uses a gateway node which acts as a cluster-based centralized network manager that sets routes for sensor data, monitors latency throughout the cluster, and arbitrates medium access among sensors. Finally they present some simulations that show the benefits of their algorithm.

Authors of [65] propose and evaluate an energy efficient clustering scheme (EECS) for periodical data gathering applications in WSNs. In the cluster head election phase, the cluster head is elected by localized competition. Further in the cluster formation phase, plain nodes join clusters using a novel distance-based method to balance the load among the cluster heads. Finally, their simulation results show that EECS prolongs the network lifetime significantly and the total energy is more efficiently consumed.

None of the previous works have used cooperative group-based network strategies in order to save energy or improve the communications in the WSN. For this reason, we presented a main idea in [22]. In this dissertation, we show the analytical model and its simulations which describe the improvement of cooperative group-based systems versus regular architectures.

2.4 Security in Group-based Topologies

Several types of security algorithms and architectures for WSNs have been developed throughout the years. In [66], Fei Hu and Neeraj K. Sharma analyzes the security challenges in wireless sensor networks and summarizes the key issues that need to be solved for achieving security in an ad hoc network. This work is a survey where the authors explain the importance to use secure mechanisms in these networks, especially for security-sensitive military applications. In [67], the authors present a study of the WSN communication security issues. Their approach is to classify the data in sensor networks, and identify communication security threats according to that classification. They propose a communication security scheme where each type of data has a different security mechanism defined.

G. Guimaraes et al. evaluate the use of security mechanisms in WSNs [7]. The authors measured the energy consumption at both radio and CPU as well as memory occupation by each of the implemented algorithms in the underlying real sensor node platforms. The authors evaluate SkipJack, RC5, RC6, TEA and DES. Their algorithms were able to check that the power consumption given by the encryption process does not cause representative impact compared with transmission energy consumption cost.

Another work, where the authors analyze the key distribution mechanisms, is presented in [68]. The authors evaluated distributed and hierarchical wireless sensor networks with unicast, multicast and broadcast communications. They estimated deterministic, probabilistic and hybrid types key pre-distribution and dynamic key generation algorithms for distributing pair-wise, group-wise and network-wise keys. Another example is [69]. The authors of this paper develop a novel group-based key pre-distribution framework. A distinguishing property of this framework is that it does not require the knowledge of sensors' expected locations and greatly simplifies the deployment of sensor networks. It can also be easily combined with any of those existing key pre-distribution techniques. After several tests, authors demonstrate that their technique can improve the security as well as the performance of existing key pre-distribution protocols substantially.

All works presented show secure algorithms for regular sensor networks. We propose a secure algorithm for a group-based network. For this reason, we will introduce some works about secure group communications and other works related with security in cluster networks. Although cluster architectures look like group-based architectures, we have seen their differences with respect to group-based topologies at the end of subsection 2.2.

B. DeCleene et al. present a hierarchical framework and key distribution algorithms for a dynamic environment [70]. They focus their research on how keys and trust relationships are transferred when users move between so-called "areas" in the hierarchy. Their preliminary analytical and simulation results indicate that it is possible to trade off communication throughput with computational and security. In this work the proposed algorithms are not intended for wireless sensor networks, therefore, these proposals are not efficient for WSN.

In [71], the authors describe the security threats and propose security services to counteract these threats in cluster-based ad hoc networks. They describe secure schemes for a mobile node to initiate, join and leave a cluster. In addition, the authors propose a security infrastructure for such networks that provides secure intra-cluster and inter-cluster communication. Finally, they also describe schemes to provide dynamic key management, member removing and member additions. This work is like the previous one, the cluster networks are formed by laptops that have better features than a sensor node.

Another work, where secure architectures for ad hoc networks are analyzed, is presented in [11]. The authors propose and evaluate architecture for secure communication in mobile ad hoc networks. Their approach divides the network into clusters and implements a decentralized certification authority. Their architecture addresses issues of authorization and access control. The proposed multi-level security model helps to adapt the complexity to the capabilities of mobile end systems.

Finally, in [12], the authors propose a novel protocol using key-distribution graph model with the Prüfer number for the secure group communications in an ad hoc network. All the mobile nodes are equipped with Global Positioning System (GPS) receivers which can get the measurements (latitude, longitude and altitude) from the GPS units. The group key is managed by using the Prüfer number and the GDH key exchange protocol. They say that this group key redistribution mode makes the security management more efficient and robust.

No one of the previous works has the same features as the one presented in this dissertation.

2.5 Synchronization in Group-based Topologies

Analyzing time synchronization topic, many protocols have been proposed for time synchronization in wireless and wired LANs. Network Time Protocol (NTP) [72] has been used on the Internet for many years. We can consider nodes on the Internet use a GPS system [73] in order to synchronize them, but this system does not work in indoor environments. Moreover, this protocol is not useful for WSNs because it consumes quite energy, CPU process and data storage and these features are limited in WSNs.

Several time protocols to synchronize nodes in WSNs have been shown lately [74]. In this paper, authors review the problem of wireless sensor network synchronization and associated key issues, including the fundamental signal processing tasks needed, such as estimation of the skew and offset between two clocks, round trip delay estimation, extension of synchronization to multiple nodes, as well as performance analysis and bounds.

One of the early works on synchronization in WSN was called post facto synchronization proposed by Elson and Estrin [75]. In this approach, each node is normally clock unsynchronized with the rest of the network, a reference node periodically broadcasts a reference broadcast message to all other nodes of the network. When an event is detected, each node stores the time received (time-stamp with its local clock). After receiving the message reference nodes will use this time as a reference and adjust their timestamps with respect to that reference. The same authors present RBS in [76]. It is a receiver-receiver protocol that requires beacon nodes that transmit reference packets (but without time stamps) to multiple nodes. Nodes record time-of-receipt and exchange these time-stamps to determine clock offsets and skews using standard linear regression techniques. However, it is not robust to clock skew, typically due to frequency offsets, and it assumes that propagation delays are negligible.

TimingSync Protocol for Sensor Networks (TPSN) [10] is based on the main idea of sender-receiver synchronization protocols. It has two phases: level discovery and synchronization. In the level discovery phase, TPSN is level-by-level time synchronization; it leads to a tree structure. In the synchronization phase, each sensor performs pairwise time synchronization with its parent in a top-down fashion. The nodes in level 2 are synchronized with each node at level 1 and so on until all nodes in the network are synchronized with respect to the root.

Flooding time synchronization protocol (FTSP [11], designed by Maroti et al., floods the whole network with messages, which contain values of the global time, i.e. the time of an elected synchronization leader. The synchronization leader periodically broadcasts synchronization messages containing its time-stamps. A network node records its local time upon reception of a synchronization message (using MAC-layer time-stamping), thus obtaining one reference point, which contains two time values: global time and local time. When a node collects an enough number of reference points, it computes the drift rate of its clock with respect to the leader clock using linear regression.

Tiny-sync and Mini-sync [12] are proposed to keep a global time in WSN by synchronizing any two nodes in the whole network. A pair of nodes uses bidirectional time-stamped packet transmissions to estimate the clock offset between them so that the two nodes are synchronized. However, since every pair of nodes must perform two-way message exchanges to get synchronized, a significant amount of communication traffic will be generated. In addition, a certain number of control packets have to be transmitted across the whole area of WSN to organize the overall network into several node-pair groups.

Also, there are time synchronization algorithms for cluster-based WSNs. In [13] H. Kim et al. propose the cluster-based hierarchical time synchronization protocol (CHTS) for wireless sensor networks. They provide network wide time synchronization through the introduction of abstractions for cluster and hierarchy. These authors introduce a novel way to calculate synchronization period to meet the required synchronization error level by using clock drift value, deterministic delays and nondeterministic offset. Finally, they simulate their algorithm in several scenarios in order to find the best environment where CHTS could work properly.

But, as we see in [77], time synchronization methods need security. There are different attacks that can provide errors in clocks for synchronizing sensor nodes. The primary goal of the attackers is to make other sensors set a wrong clock time. The secondary goal of the attackers is to affect more sensors by making themselves locate close to the root of the sync-tree, i.e., having low levels. Therefore synchronization protocols should be safe. In the same paper [77], the authors propose a scheme with three phases: a) abnormality detection performed by individual sensors, b) trust-based malicious node detection performed by a base station, and c) self-healing through changing topology of synchronization. This solution is tested under two attacks and the conclusions are that their system can quickly detect and defeat the misleading attack and the wormhole attack, with reasonable implementation overhead tree.

Another example is [78]. Authors propose a suite of protocols for secure pairwise and group synchronization of nodes that lie in each other's power ranges and of nodes that are separated by multiple hops. Their protocols offer different points of operation in the energy-accuracy subspace and the choice of the specific protocol should be made by the network designer depending on his application needs. Finally, [79] is another paper where a secure clock synchronization process is proposed. The idea behind their scheme is using node's neighbors as verifiers to check if the synchronization is processed correctly so that it can detect the attacks launched by compromised node. This scheme is based on three phases: a) Level Discovery Phase, b) Synchronization Phase and c) Verification Phase. According to their simulations, this scheme guarantees that normal nodes can synchronize their clocks to global clock even if each normal node has up to colluding malicious nodes during synchronization phase.

2.6 Group-based Applications Environment

Group-based networks can be used when it wants to setup a network where groups could appear and join the network at any time or when the network has to be split into smaller zones to support a large number of nodes, that is, in any system where the devices are grouped and there must be connections between groups.

The following list gives several group-based WAHSN application areas:

1. Let us suppose a job where all human resources need to be split into groups to achieve a purpose (such as fire fighter squads for putting out the fire). Now, let's suppose that all people involved in that activity need a device that has to be connected with other devices in the same group to receive information from the members within the group, and closer groups have to be connected to coordinate their efforts. Currently coordination between groups is done through a wireless connection to the command center or using satellite communications. But, sometimes neither of those solutions can be used because a free of obstacles line of sight is needed, because there are too many wall loses or because more gain or power is needed to reach the destination.

2. For battle field communication, it is especially useful for inter-squad communication to collaborate when an objective is targeted by position detectors.
3. Groups could also be established because of geographical locations or unevenness. It happens in rural and agricultural environments. A group-based topology in this kind of environment could be useful to detect plagues or fire and to propagate an alarm to neighbor lands. It will provide easier management and control for detecting fires and plagues as well as allowing scalability.
4. Health monitoring. A patient could need to be monitored in several locations while he is doing his activity. Every room or place could have a group of sensors (and even each group with different type of topology inside) and neighbor groups must be communicated to keep track of the patients.
5. It could be used in any kind of system in which an event or alarm is based on what is happening in a specific zone, but conditioned to the events that are happening in neighbor zones. One example is a group-based system to measure the environmental impact of a place. It could be better measured if the measurements are taken from different groups of sensors, but those groups of sensors have to be connected in order to estimate the whole environmental impact.
6. Group-based virtual games. There are many games where the players are grouped virtually in order to perform a specific task. Interactions between groups in virtual reality should be given by interactions between players from different groups to exchange their knowledge.

2.7 Conclusion

The main goal in a wireless group-based topology is the network protocol and the group management, that is the design of an efficient algorithm for a new node to find nearest (or the best) group to join in it.

We have shown a detailed state of the art of wireless sensor networks. Firstly, we make an introduction about topologies in WSNs, particularly we have focused our attention to see the differences between the existing architectures, such as cluster networks, LANMAR, etc. and our proposal. Moreover, we have introduced several related works about collaboration in networks, and how this feature improves the efficiency of the network. Security in wireless sensor networks, we have focused our attention in some works where their authors apply security in group topologies. Then, we have explained several related work about synchronization topic, which is very important in this kind of networks. Finally we have shown some applications where group-based topologies could be applied.

Finally, work presented in this chapter has been published in the following references [37], [31], [32], [80], [81], [34] and [33].

Chapter 3

GROUP-BASED TOPOLOGIES PERFORMANCE STUDY

3.1 Introduction

The first issue that a researcher has to know is if groups and/or clusters improve or not the global behavior of a network. The authors of [82] take a closer analytical look at WSNs of clustered organization. They prove that these networks do not necessarily outperform non-clustered WSNs. The condition that ensures superior performance of clustered WSNs, with absolute certainty, is that the formed clusters lie within the isoclusters of the monitored phenomenon. Based on the operational cost results presented on the paper, the content criterion of the optimal WSN clustering strategy requires clusters committed to neighboring nodes with highly correlated data-readings. Finally, they show that 5-hop clusters, in which the cluster head is at most 2 hops away from any node in the cluster, can ensure near-optimal network performance under a wide range of cluster-to-sink and cluster-to-isocluster spatial arrangements.

This section compares the performance of 3 common routing protocols used in MANET and it shows which one is the best when they are using group-based topologies.

3.2 Test Bench

First, we present the test-bench used for all the evaluated protocols. The number of nodes and the coverage area of the network have been varied. We have simulated 4 scenarios for each protocol: the first one with fixed nodes; the second one with mobile nodes and failures; the third one with grouped nodes; and, the fourth one with grouped mobile nodes and failures. We have simulated each scenario for 100 and 250 nodes to observe the system scalability. It has been obtained using the version Modeler of OPNET simulator [83].

Instead of a standard structure we have chosen a random topology. The nodes can move randomly during the simulation. The physical topology does not follow any known pattern. The obtained data do not depend on the initial topology of the nodes or on their movement pattern, because all of it has been fortuitous. In order to take measurements from the mobile nodes simulation, we have forced failures in the networks with the consequent recovering processes. It allows us to observe the network behavior, against physical topology changes and node failures. Failures and recoveries usually happen in these kinds of networks, so, we are going to study how a network-level protocol works when those events occur.

We have created 6 groups for the 100 nodes topology, covering approximately, a circular area with a 150 meter radius each group. There are approximately 16 or 17 nodes in each group. The number of nodes in each group varies because of the node’s random mobility. A node can change from one group to another at any time. For the 250 nodes topology, we have created 12 groups, with 15 or 16 nodes per group approximately, covering a circular area with a 150 meter radius each group. The ad-hoc nodes of the topologies have a 40 MHz processor, a 512 KB memory card, a radio channel of 1 Mbps and their working frequency is 2.4 GHz. Their maximum coverage radius is 50 meters. This is a conservative value because most of the nodes in ad-hoc network have larger coverage radius, but we preferred to have lower transmitting power for the ad-hoc devices to enlarge their lifetime. The traffic load used for the simulations is MANET traffic generated by OPNET. We inject this traffic 100 seconds after the beginning. The traffic follows a Poisson distribution (for the arrivals) with a mean time between arrivals of 30 seconds. The packet size follows an exponential distribution with a mean value of 1024 bits. The injected traffic has a random destination address, to obtain a simulation independent of the traffic direction. We have simulated both scenarios for DSR, AODV and OLSR protocols. The results obtained are shown in the following subsections.

3.3 Average Delay at Application Layer

Figure 3-1 and **Figure 3-2** show the average delay of the DSR protocol in fixed and mobile topologies at the application layer. In figure 1 we observe that group-based topologies have an average delay close to 0.005 seconds regardless of the number of nodes in the network. In the regular network the delay has a value of 0.02 seconds for 100 nodes topology and of 0.03 seconds for the 250 nodes topology when the network converges. In the case of the 100 nodes topology there is an improvement of 75% and it is better in the 250 nodes topology (an 83% improvement). The topologies with mobility and errors (figure 2) shows that the average delays at the application layer are higher in the group-based topologies until the network converges. We observe that group-based topologies present worse behavior up to 1300 seconds. Then, the delay decreases. There is an improvement of around 5%.

The average delay at the application layer in the AODV protocol can be seen in **Figure 3-3** and **Figure 3-4**. When we are talking about fixed topologies (**Figure 3-3**), both 100-nodes and 250-nodes, give an average delay higher than 0.5 seconds when the network converges, but there are some peaks higher than 2.5 seconds. On the other hand, group-based topologies have a similar delay which is around 0.15 seconds. Group-based topologies improve the delay at the application layer by 70%.

When the topology with mobile nodes is used, the simulation shown in figure 4 is obtained. In the case of 250 nodes, there is a delay of 1 second when the network has converged. The case of 100 nodes gives an average delay around 0.75 seconds. When there are group-based topologies, the delay decreases to 0.25 seconds in both cases. There is an improvement of 75% for the 250-nodes topology and 67% for the 100-nodes topology.

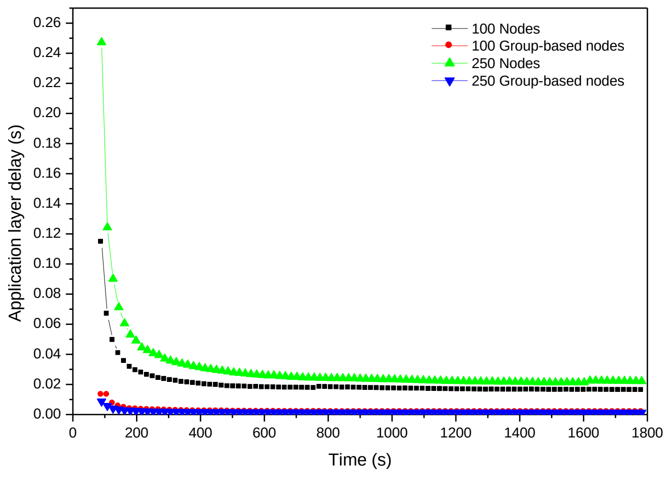


Figure 3-1: DSR average delay at application layer in fixed topologies.

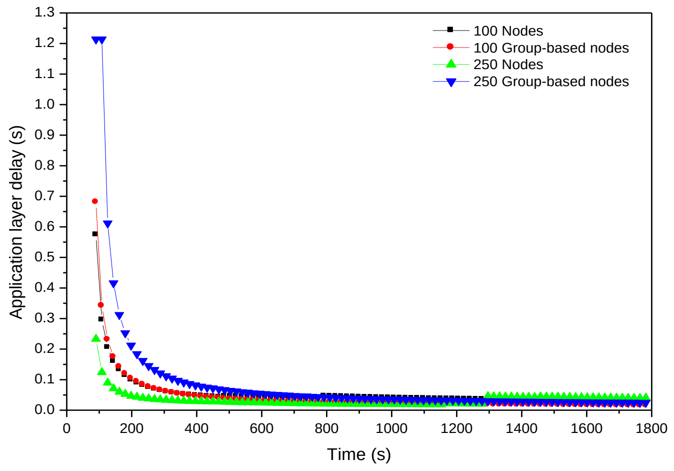


Figure 3-2: DSR average delay at application layer in mobile topologies.

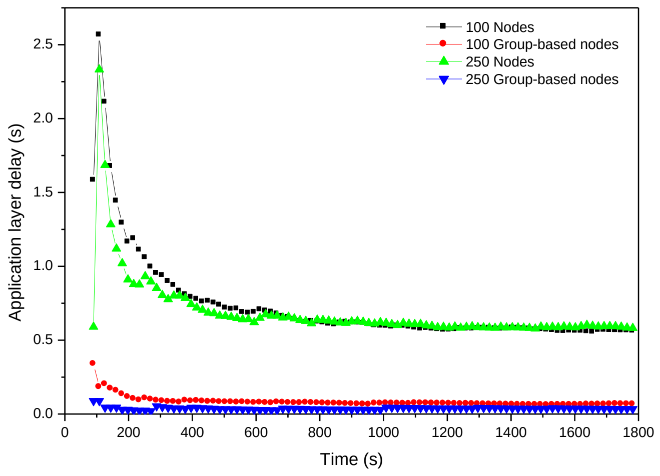


Figure 3-3: AODV average delay at application layer in fixed topologies.

In **Figure 3-5**, the delay at the application layer for the OLSR protocol using fixed topologies is shown. In the case of 250 nodes we have obtained a delay of around 0.015 seconds, which has changed to 0.0035 seconds in the case of 250-nodes group-based topology (there is a 76% improvement). In the case of 100 nodes, it has decreased from 0.005 seconds in the regular topology to 0.002 seconds in the group-based topology, so there is a 60% improvement.

When there is mobility, errors and failures in the network for the OLSR protocol (see **Figure 3-6**), we observe that the 100-nodes regular topology has a delay at the application layer of 0.007 seconds when the network has converged, but there is a delay of 0.0025 seconds for the 100-nodes group-based topology (a 64% improvement). In the case of 250 nodes the improvement is around 60 %. We have obtained a delay of 0.005 seconds in the regular topology versus 0.002 seconds in the group-based topology.

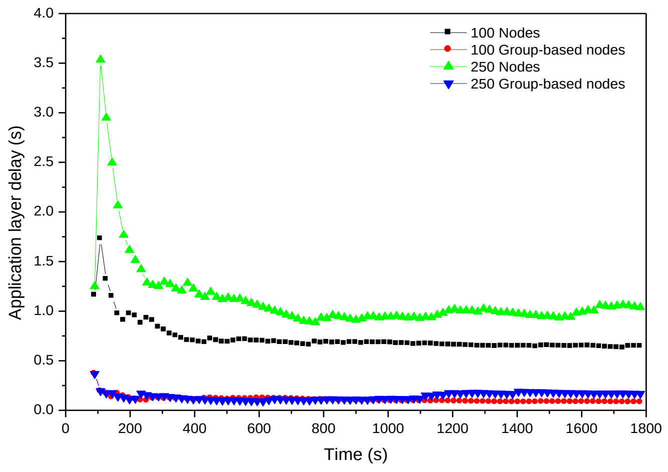


Figure 3-4: AODV average delay at application layer in mobile topologies.

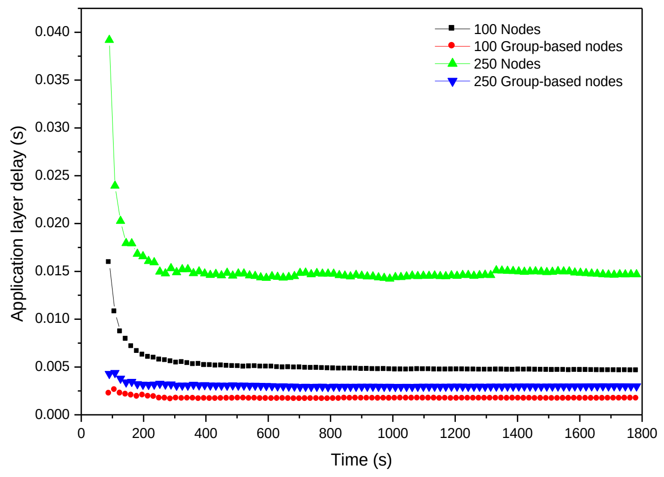


Figure 3-5: OLSR average delay at application layer in fixed topologies.

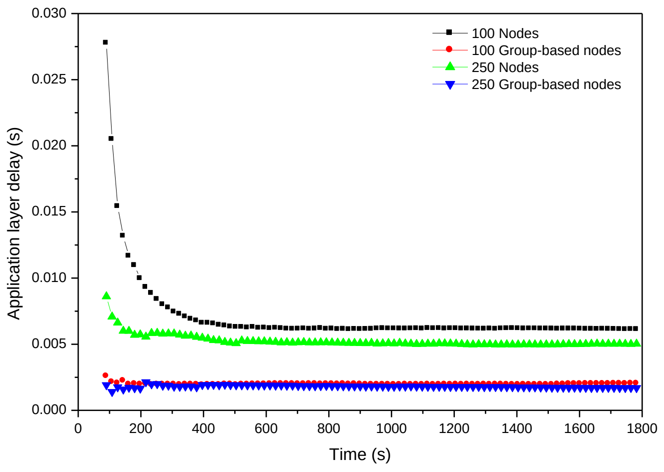


Figure 3-6: OLSR average delay at application layer in mobile topologies.

3.4 Routing Traffic Received

We have compared the routing traffic received in the DSR protocol (**Figure 3-7** and **Figure 3-8**). **Figure 3-7** shows that the traffic is quite stable because it is a fixed network without errors or failures. The traffic received in the 250-node topology is around 500 Kbits/s, but when we group the nodes, this traffic decreases to 200 Kbits/s (a 60% improvement). The value obtained in a 100-node topology (250 Kbits/s), is also improved when we group the nodes (100 Kbits/s), therefore there is a 60% improvement.

In **Figure 3-8** we observe a similar behavior. In this case we conclude that when there are errors and failures in the 250-nodes topology the traffic fluctuates and is less stable (we can observe it in the intervals from 600 to 800 seconds and around 1200 seconds). We also observe that the instability is much lower in group-based topologies. 100-nodes topology has a mean value around 175 Kbits/s, while 100-nodes group-based topology has a mean value around 95 Kbits/s, so there is an improvement of 46%. On the other hand, 250-nodes topology has a mean value around 400 Kbits/s, while 250-nodes group-based topology has a mean value around 180 Kbits/s, so there is an improvement of 55%.

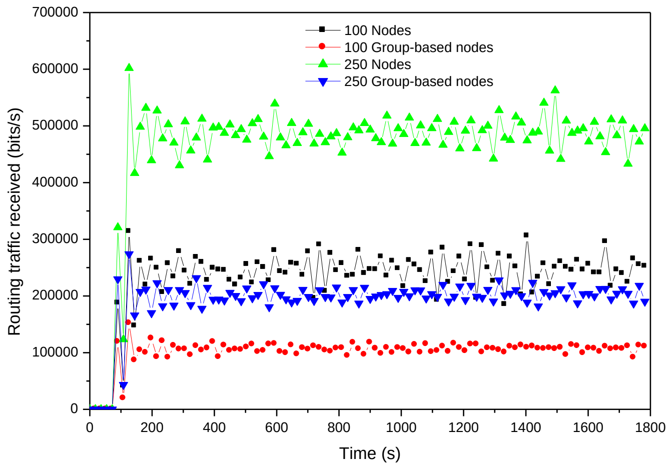


Figure 3-7: DSR routing traffic received in fixed topologies.

Then, the routing traffic received for the AODV in each simulated topology can be seen in **Figure 3-9** and **Figure 3-10**. We observe that the routing traffic received is independent of the mobility of the nodes. In **Figure 3-9** we can see that the routing traffic goes from 440 Kbits/s for 250-node case to 250 Kbits/s when there are group of nodes (a 43% improvement). In the 100-node topology, it goes from 230 Kbits/s to 140 Kbits/s in the group-based topology case (a 39% improvement). When there are mobility, errors and failures (see **Figure 3-10**) in the 250-node topology, the values go from 440 Kbits/s to 250 Kbits/s in the group-based topology (a 43% improvement). We obtained 200 Kbits/s in the regular 100-node topology and 135 Kbits/s for the group-based one (a 32% improvement).

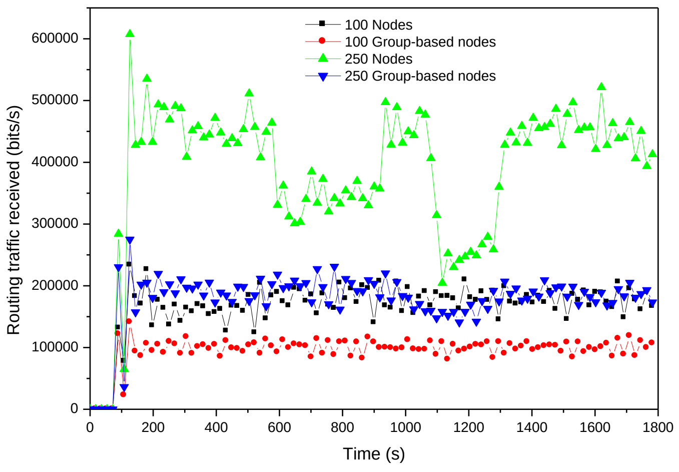


Figure 3-8: DSR routing traffic received in mobile topologies.

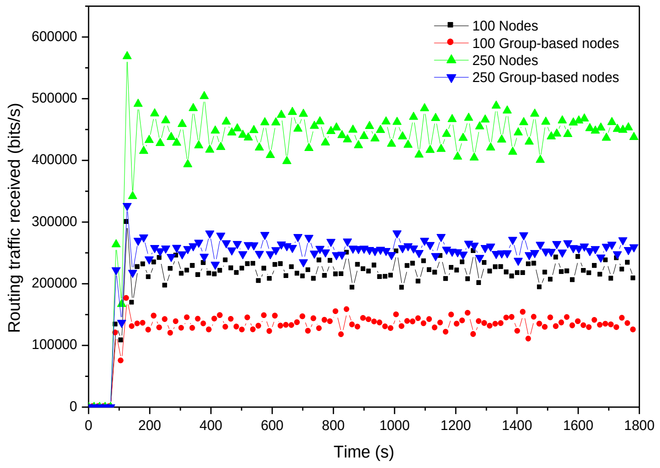


Figure 3-9: AODV routing traffic received in fixed topologies.

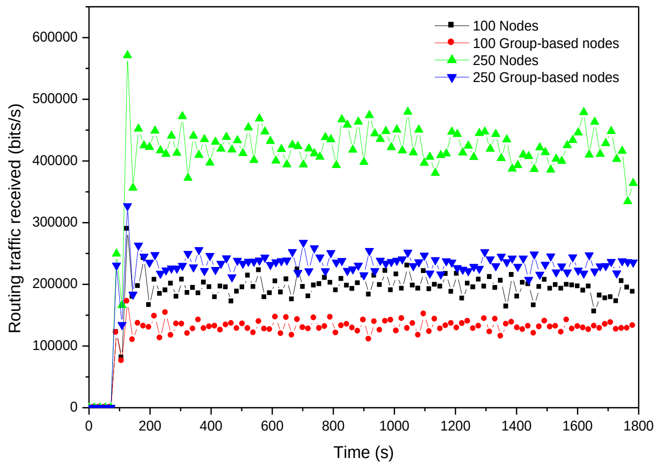


Figure 3-10: AODV routing traffic received in mobile topologies.

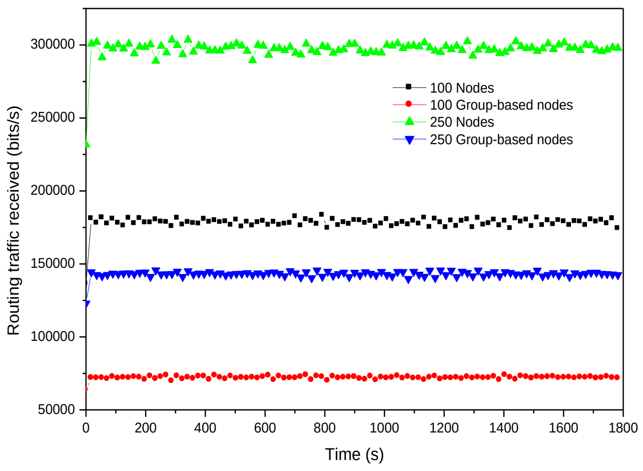


Figure 3-11: OLSR routing traffic received in fixed topologies.

Finally, we have studied the behavior of OLSR protocol analyzing the mean routing traffic received (**Figure 3-11** and **Figure 3-12**). In **Figure 3-11**, we see that the routing traffic received in the 100-node fixed topology was around 180 Kbits/s, while in group-based topology it has decreased to 70 Kbits/s, so there is a 61% improvement. In the 250-node topology case, we appreciate that this traffic was approximately 300 Kbits/s, but there are values lower than 150 Kbits/s in the group-based topology, so there is a 50% improvement. **Figure 3-12** shows the results of a network with mobility and errors and failures. We have observed some fluctuations due to failures and errors in the network, in both topologies. Those fluctuations are minimized when we use group-based topologies. Improvements of 61% and 50% are obtained in both topologies, respectively.

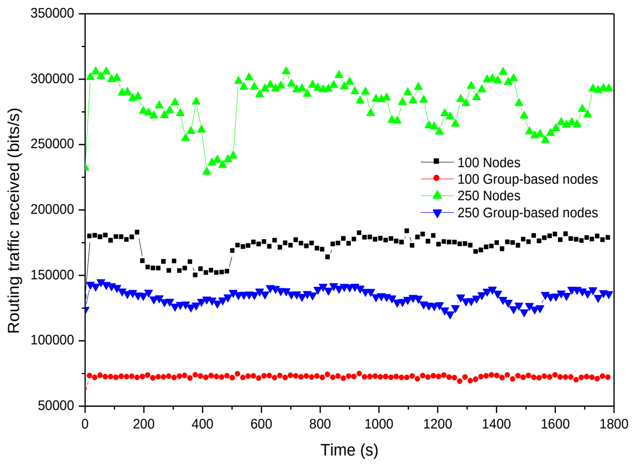


Figure 3-12: OLSR routing traffic received in mobile topologies.

3.5 Throughput

When we study the network throughput (**Figure 3-13** and **Figure 3-14**), we observe that group-based topologies give a much lower value than the one obtained in regular topologies.

For the 100-node topology (**Figure 3-13**), the throughput varies from 225 Kbits/s to 100 Kbits/s in the group-based topology (a 56% improvement). In the 250-node topology we obtain 460 Kbits/s of throughput for the regular topology and 190 Kbits/s of throughput for the group-based one (a 59% improvement). Moreover, when we compare **Figure 3-13** and **Figure 3-14**, we can conclude that the throughput in group-based topologies has a very low variation regarding a fixed or mobile scenario. The obtained improvement is quite important. We can see in **Figure 3-14** that, after 1200 seconds, the obtained throughput in 250-node topology is similar to the obtained throughput in the 100-node topology.

Figure 3-15 shows throughput for fixed topologies. The 100-node scenario gives a 200 Kbits/s mean value, but a value of 120 Kbits/s is obtained for the group-based scenario (a 40% improvement). In the 250-node case, we obtain mean values of 425 Kbits/s for the fixed scenario and of 225 Kbits/s for the group-based scenario (a 47% improvement). **Figure 3-16** shows the results for mobile topologies with errors and failures. The improvement obtained by grouping nodes, it decreases in the 100-node case (37%), but it doesn't vary in the 250-node cases.

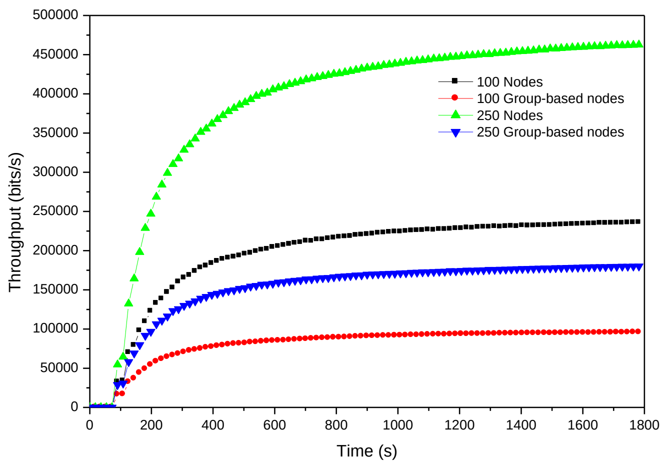


Figure 3-13: DSR mean throughput in fixed topologies.

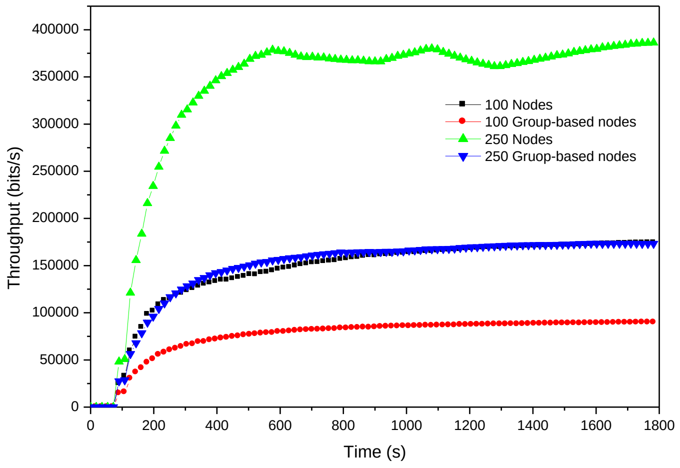


Figure 3-14: DSR mean throughput in mobile topologies.

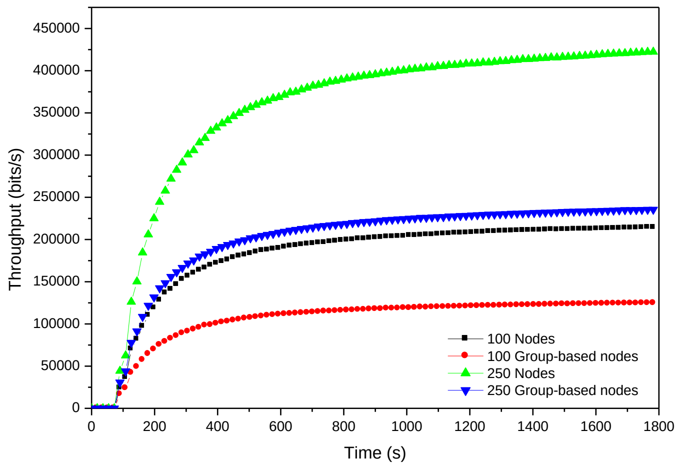


Figure 3-15: AODV mean throughput in fixed topologies.

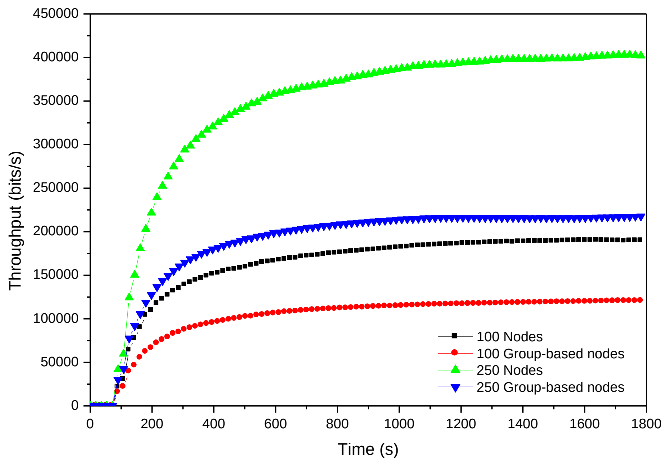


Figure 3-16: AODV mean throughput in mobile topologies.

Finally, average throughput measured in fixed topologies can be observed in **Figure 3-17**. In scenarios with 250 nodes we obtained throughputs of 550 Kbits/s and 250 Kbits/s (group-based, with a 54% improvement). In 100-node regular topology the throughput is 325 Kbits/s and 125 Kbits/s (group-based, with a 61% improvement). When we consider mobility, errors and failures (**Figure 3-18**) the throughput is not as stable as in above case but, we can observe that improvements are quite similar. In case of 250 nodes we obtain a 52% improvement in group-based scenario; in case of 100 nodes the improvement reaches the 60%.

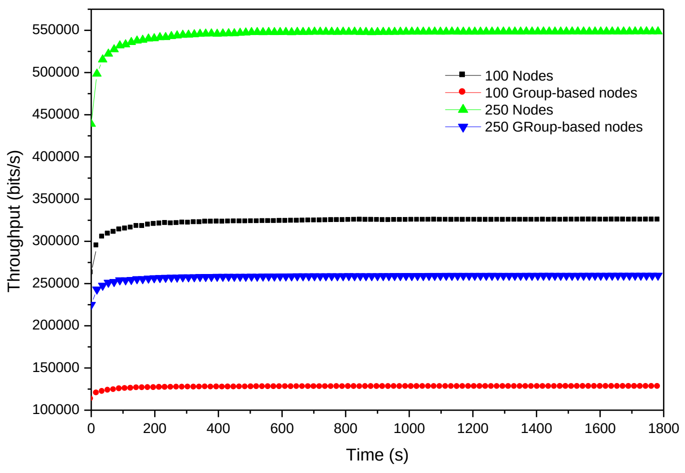


Figure 3-17: OLSR mean throughput in fixed topologies.

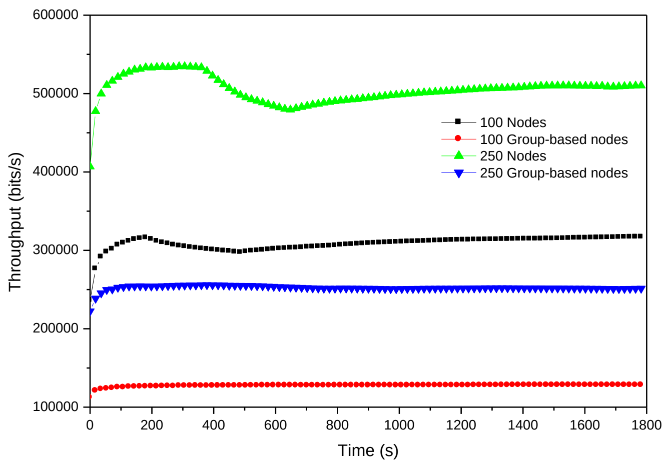


Figure 3-18: OLSR mean throughput in mobile topologies.

3.6 Group-based Topologies Comparison

In order to make the comparison of DSR, AODV and OLSR using group-based topologies, we have used the same test bench used previously. This comparison will show us which mobile and ad-hoc routing protocol performs better using group-based topologies.

Figure 3-19 shows the average delay at application layer in fixed group-based topologies. The most instable protocol and with higher delay in 100-nodes and 250-nodes topologies is AODV protocol. It has peaks with more than 0.45 seconds and it is stabilized around 1700 seconds with a mean value of 0.15 seconds. DSR and OLSR are the ones with lowest delay. **Figure 3-20** shows the average delay at application layer in mobile topologies. DSR protocol is the one that has worst delay until the network converges. Then, when the network is stabilized, the worst is AODV protocol which has delays between 0.1 and 0.15 seconds. OLSR protocol gives the lowest delays.

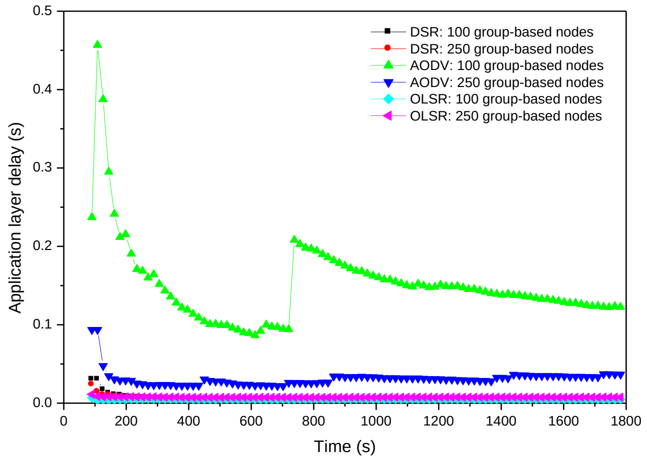


Figure 3-19: Comparison of delay at application layer in fixed topologies.

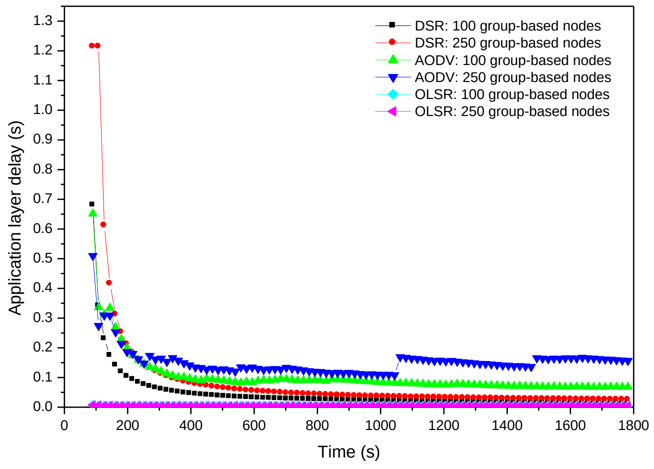


Figure 3-20: Comparison of delay at application layer in mobile topologies.

The routing traffic received in fixed and mobile group-based topologies is shown in **Figure 3-21** and **Figure 3-22** respectively. In fixed group-based topologies (see **Figure 3-21**) AODV protocol is the one that gives higher routing traffic received (around 250 Kbit/s in 250-node topology and 135 Kbits/s in 100-nodes topology). OLSR protocol is the most stable and the one with lower routing traffic received (145 Kbits/s in 250-node topology and 70 Kbits/s in 100-node topology). When the mobile group-based topologies are analyzed (**Figure 3-22**), AODV protocol is the one that has worst behavior and OLSR is the most stable and the one that has lower routing traffic sent. DSR protocol is the most instable.

The average throughput consumed in the fixed group-based topologies is compared in **Figure 3-23**. The protocol that consumes the lowest throughput is the DSR protocol (90 Kbits/s in the 100-node topology and 170 Kbits/s in the 250-node topology). The protocol with the most stable throughput consumed is the OLSR protocol. When the network converges, both AODV and OLSR protocols have the same average throughput in the 100-nodes topology, but the OLSR protocol has the lowest convergence time. In case of having a group-based topology with mobility, errors and failures (see **Figure 3-24**). Results are very similar to the previous ones. The protocol that consumes lower throughput is DSR. AODV protocol consumes lower throughput while the network is converging, but this throughput becomes very similar to the one given by OLSR protocol when the network converges. OLSR protocol is still the most stable.

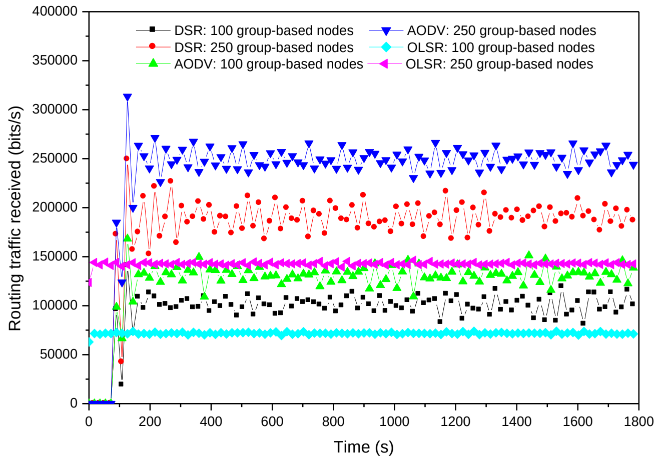


Figure 3-21: Comparison of routing traffic received in fixed topologies.

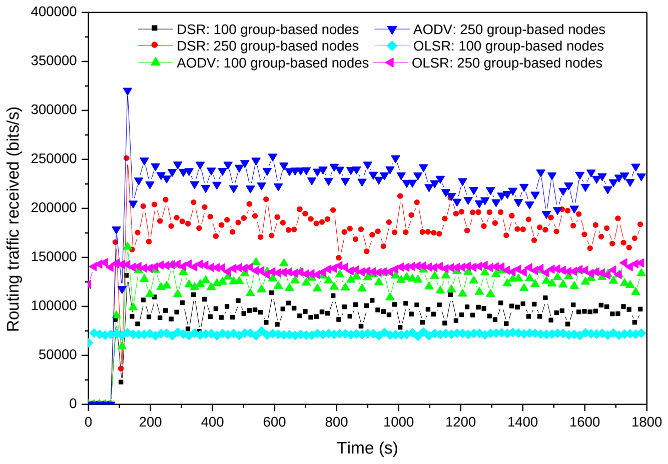


Figure 3-22: Comparison of routing traffic received in mobile topologies.

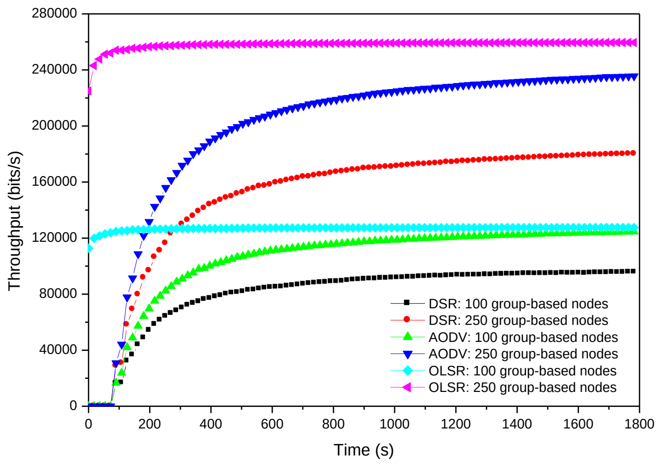


Figure 3-23: Comparison of average throughputs consumed in fixed topologies.

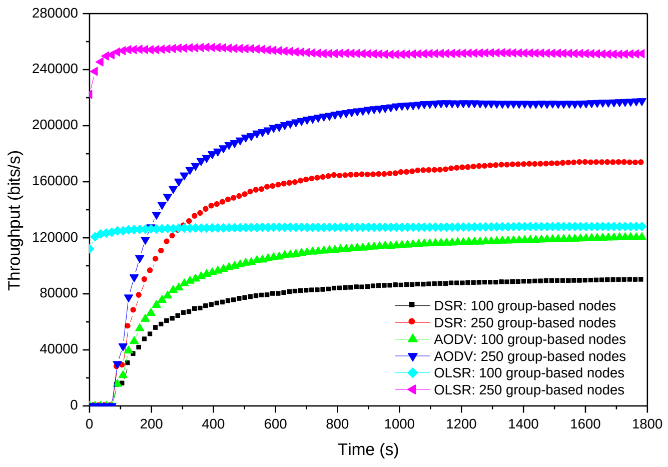


Figure 3-24: Comparison of average throughputs consumed in mobile topologies.

The average delay at MAC layer in fixed group-based topologies is shown in **Figure 3-25**. All routing protocols have an average delay lower than 0.001 seconds when the network has converged in both 100-nodes and 250-nodes topologies. It shows that group-based topologies have a good behavior. DSR protocol with 100-node topology has been the one with worst behavior and OLSR in 250-node topology has been the best one. OLSR protocol has the same delay (around 0.001 seconds) for both topologies, 100-nodes and 250-nodes, approximately, and it is the most stable. **Figure 3-26** shows the simulation for mobile and errors and failures topologies. All protocols have a delay lower than 0.001 seconds when the network has converged. In this case, AODV protocol has the worst behavior and OLSR protocol is the most stable.

Now we have analyzed the route request sent in reactive protocols for fixed and group-based topologies (**Figure 3-27** and **Figure 3-28** respectively). AODV protocol is the one with most number of route requests sent (860 approximately in 250-nodes topology and 330 approximately in 100-nodes topology). We have observed a relationship between the number of route requests sent in the AODV protocol and number of nodes in the topology. There is approximately a factor of 3.3. In the DSR protocol, the number of route requests sent is equal to 730 in the 250-nodes topology and 190 in the 100-nodes topology. Both, fixed and mobile, present the same behavior. We have observed that the route request sent is the only parameter that gives worst values in group-based topologies than in regular topologies.

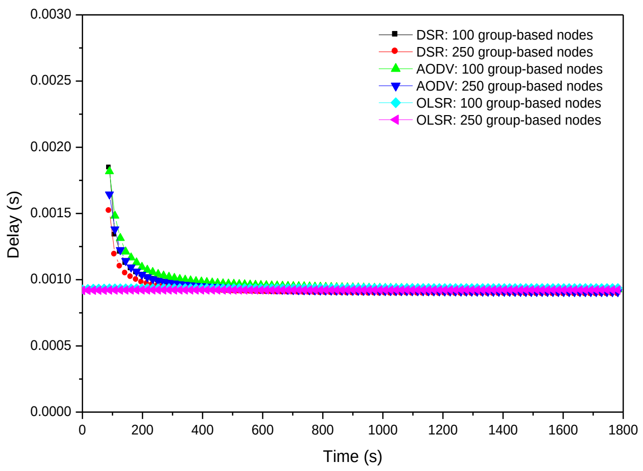


Figure 3-25: Comparison of average delays at MAC layer in fixed topologies.

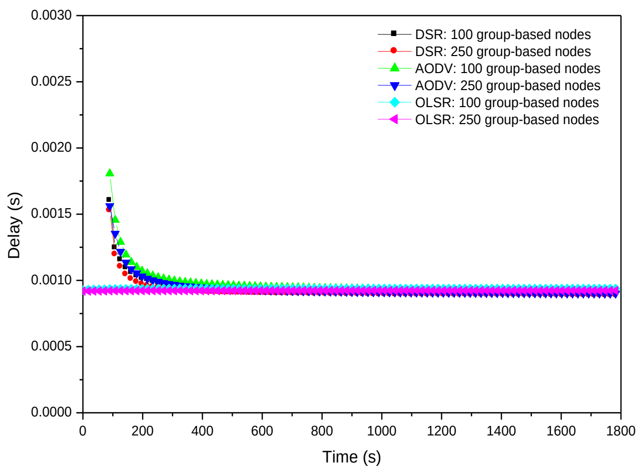


Figure 3-26: Comparison of average delays at MAC layer in mobile topologies.

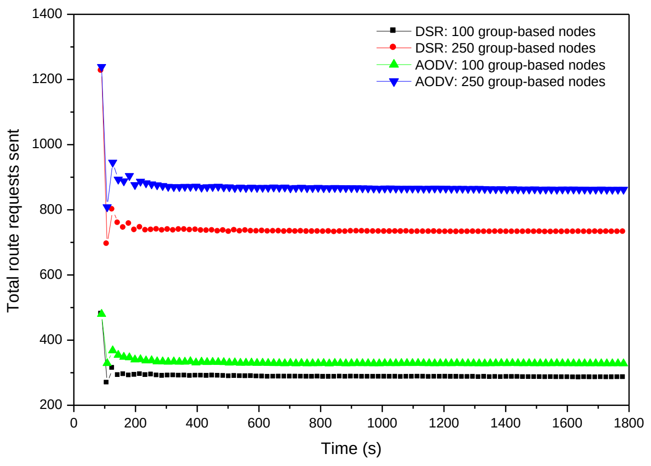


Figure 3-27: Comparison of route request sent in fixed topologies.

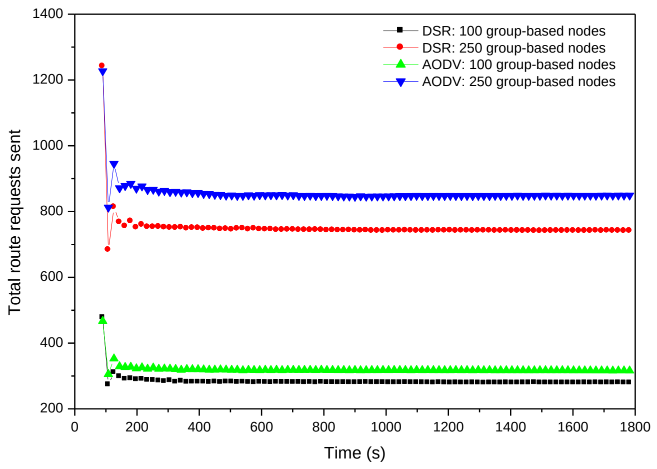


Figure 3-28: Comparison of route request sent in mobile topologies.

3.7 Analyzed Protocols Summary

In this subsection we show the benefits of using a group-based topology in ad-hoc networks and we show several examples in which they can be used. We have simulated DSR, AODV and OLSR protocols with and without groups and the results show that group-based topologies give better performance. In **Table 3-1** we can see a summary where there is percentage improvement when group-based topologies are used.

In this study we have made other measures. **Table 3-2** shows the best and worst protocols for every one of the parameters analyzed. The best improvement percentage, when group-based topologies are used, was the DSR protocol when the average delay at the application layer was simulated. On the other hand, in the same case for mobile topologies, DSR protocol gave the worst percentage of improvement.

We observed more percentage of improvement in fixed topologies when there are more nodes in the topology, but when there is a mobile topology, the improvement is higher in the topology with lower number of nodes. We have also observed that when a routing protocol is the best one in a fixed group-based topology, it continues being the best one in the mobile group-based topology. On the other hand, we observed that a routing protocol, which is the best (or worst) in a group-based fixed topology, could not be the best (or worst) in the mobile topology. The routing protocol that appeared as the best one was OLSR and the one that appeared as the worst was AODV.

Table 3-1: Percentage of improvement when group-based topologies are used.

	Fixed topology (100 nodes)	***Fixed topology (250 nodes)***	***Mobile Topology (100 nodes)***	***Mobile Topology (250 nodes)***
DSR Average delay at the application layer	75%	83%	5%	5%
DSR Routing traffic received	60%	60%	46%	55%
DSR mean throughput	56%	59%	48%	55%
AODV Average delay at the application layer	70%	70%	67%	75%
AODV Routing traffic received	39%	43%	32%	43%
AODV mean throughput	40%	47%	37%	47%
OLSR Average delay at the application layer	60%	76%	64%	60%
OLSR Routing traffic received	61%	50%	61%	50%
OLSR mean throughput	54%	61%	52%	60%

Table 3-2: Comparison of routing protocols in group-based topologies.

	Best in fixed topology	***Best in mobile topology***	***Worst in fixed topology***	***Worst in mobile topology***
Delay at MAC layer	OLSR	OLSR	DSR	AODV
Throughput consumed	DSR	DSR	AODV/OLSR	AODV/OLSR
Application traffic	AODV	DSR	OLSR	OLSR
Routing traffic sent	OLSR	OLSR	AODV	AODV
Routing traffic received	OLSR	OLSR	AODV	AODV
Delay at application layer	DSR/OLSR	OLSR	AODV	AODV
Average number of hops in a path	AODV	DSR	AODV	DSR
Route request sent	DSR	AODV	DSR	AODV

3.8 Conclusion

In this chapter we have simulated DSR, AODV and OLSR protocols with and without groups and the results show that group-based topologies give better performance for wireless ad-hoc networks. So, grouping nodes increases the productivity and the performance of the network with low overhead and low extra network traffic. Therefore, good scalability can be achieved in group-based networks. On the other hand, the protocol that gives better results has been OLSR because this protocol introduces less routing traffic and it has behavior more regular.

The best improvement percentage has been the DSR protocol when the average delay at the application layer has been simulated. We have observed more improvement in fixed topologies when there are 250 nodes in the topology, but when there is a mobile topology, the improvement is higher in the topology with 100 nodes. When a routing protocol is the best one in a fixed group-based topology, it continues being the best one in the mobile group-based topology. On the other hand, we have observed that a routing protocol, which is the best (or worst) in a group-based fixed topology, could not be the best (or worst) in the mobile topology. The routing protocol that has appeared more as the best one has been OLSR and the one that has appeared as the worst one has been AODV.

Finally, work presented in this chapter has been published in the following references [16], [14], [58] and [20].

Chapter 4

GROUP-BASED ARCHITECTURE AND PROTOCOL FOR WSNS

4.1 Introduction

First, a distinction has to be presented between a groupware architecture, where all nodes collaborate towards the proper operation and the success of the purpose of the network (despite of where they are placed and the relationships between them), and group-based architecture, where the whole network is broken down into groups and each group could perform different operations (but information could be transmitted between groups and there must exist connections between nodes from different groups). In a wireless group-based architecture, a group consists of a set of nodes that are close to each other (in terms of geographical location or in terms of round trip time). These groups could have a link if a node from a group is near a node from another group. The distance between two nodes is given by the network latency or the round trip time.

The main goal in a wireless group-based topology is the network protocol and the group management, that is the design of an efficient algorithm for a new node to find nearest (or the best) group to join in it. Then, some important issues must be designed despite of the routing protocol inside each group (it could be different for each group): (i) How to build neighboring groups, and (ii) a protocol to exchange messages between neighboring groups.

The performance of the group-based network highly depends on the efficiency of this nearby group locating process. All works found in the literature, where nodes are divided into groups and connections are established between nodes from different groups, have been developed to solve specific issues such as distribution service in multimedia networks [42], group-based games over NAT/Firewalls [41], and so on.

Two types of group-based topologies can be distinguished, as we have seen in Chapter 2:

- Planar group-based topologies.
- Layered group-based topologies.

There are several differences between both group-based topologies. While layered group-based topologies grow structured organized by upper layers, planar group-based topologies grow unstructured without any organization. In layered group-based topologies anyone can know exactly where each group is and how to reach it. Otherwise planar group-based topologies, every time a node wants to reach other group, the message should travel through many unknown groups due to groups join the network as they appear. Delays between groups in layered group-based topologies could be lower because connections between groups can be established taking into account this parameter, otherwise, in planar group-based topologies connections between groups are established by group's position, their geographical situation or because of their appearance in the network.

Layered networks address several complexities because nodes could have several types of roles and fault tolerance for every layer must be designed. On the other hand, planar networks are more simplex because all nodes have the same role. In order to have scalability, layered group-based topologies must add more layers to its logical topology, while planar group-based topologies could grow without any limitation, just the number of hops of the message.

Cluster-based networks are a subset of the group-based networks, because each cluster could be considered as a virtual group. But a group-based network is capable of having any type of topology inside any group, not only clusters. In cluster based architectures each cluster has adjacencies with other clusters and all clusters have the same rules. A cluster has three types of nodes: a) Cluster Head node, b) Cluster Gateways and c) Cluster Members [84] [85]. Cluster Head manages its cluster, and it routes the information to other clusters, in order to send the data to the correct destination. The rest of the nodes in a cluster are cluster members. This kind of networks usually has a size of 1 or 2 hops from the Cluster Head node. A cluster member does not have inter-cluster links.

On the other hand, other works of groups systems such as the following will not be considered in this dissertation:

1. The community based mobility model for ad hoc network research presented in [53], because although the network is organized in groups, and nodes can move from one host to another, there is not any connection between border nodes from different groups.
2. The landmark hierarchy presented in [54] because although there is a node with higher role which has connections with nodes from other groups, its leaf nodes do not have.
3. Another example similar to the last one is the BGP routing protocol architecture [55]. The routers inside the autonomous system do not have connections with routers from other autonomous systems unless they are BGP routers. They are the backbone of the network, so they are in the highest hierarchy level.
4. It is not considered moving groups such as Landmark Routing Protocol (LANMAR [56]), where the set of nodes move as a group, so the group can enlarge or diminish with the motion of the members.

5. Multicast groups for ad-hoc and sensor networks [86], because there are not connections between groups.

In this chapter we are going to show our group-based proposal. This proposal is defined by two elements. Firstly, an architecture that defines types of nodes and their features, connections between nodes, etc. Then, inside this architecture we analyze how a node selects a neighbor and how the number of nodes varies according to network topology. Furthermore, we will show the messages needed to carry out this group-based architecture. Secondly, we will describe the operations of our group-based protocol in order to manage our groups in a WSN. Finally, we will present some connection time simulations to show the time needed to send information from a group to another and we will show our conclusions about this chapter.

4.2 Group-based Architecture Operation

An architecture, where the structure of nodes is based on the creation of groups and the nodes have the same functionality in the network, is proposed. In each group exists a head node that limits the zone where the node from the same group will be placed, but its functionality is the same that the rest of the nodes in the network. This head node is different than the head nodes of cluster networks because a head node in a cluster network has a total control over the cluster. A head node is our architecture only helps to create each group, and it is changing continually in order to be as centered as possible in a group.

Every node has a node identifier (called *nodeID*) that is unique in its group. The first node in the network acquires a group identifier (called *groupID*) that could be given doing a simple one-way operation with its *nodeID*. New joining nodes (normally leaf nodes) will know their group identifier from their new neighbors. End nodes are, physically, the edge nodes of the group. When there is an event in one node, this event is sent to all the nodes in its group in order to take the appropriate actions. Moreover, there are other nodes called eligible leaf nodes, which can connect to other groups, too. Nest, these nodes will be explained. **Figure 4-1** shows an example of our proposed architecture topology.

All groups could have the same or different routing protocol inside. All nodes in a group can know all information about their group using the routing protocol (see references [4] and [87] as examples). End nodes have connections with other end nodes from neighbor groups and are used to send information to other groups or to receive information from other groups and distribute it inside. Because it is wanted a fast routing protocol, OLSR [88] has been chosen to route information inside all groups, but it can be changed by other routing protocol depending on the network’s characteristics. When the information is for a node of the same group it is routed using the *nodeID*. Every node runs its link state database locally and selects the best path to a destination based on a metric (described later). All these steps will be done by the routing protocol.

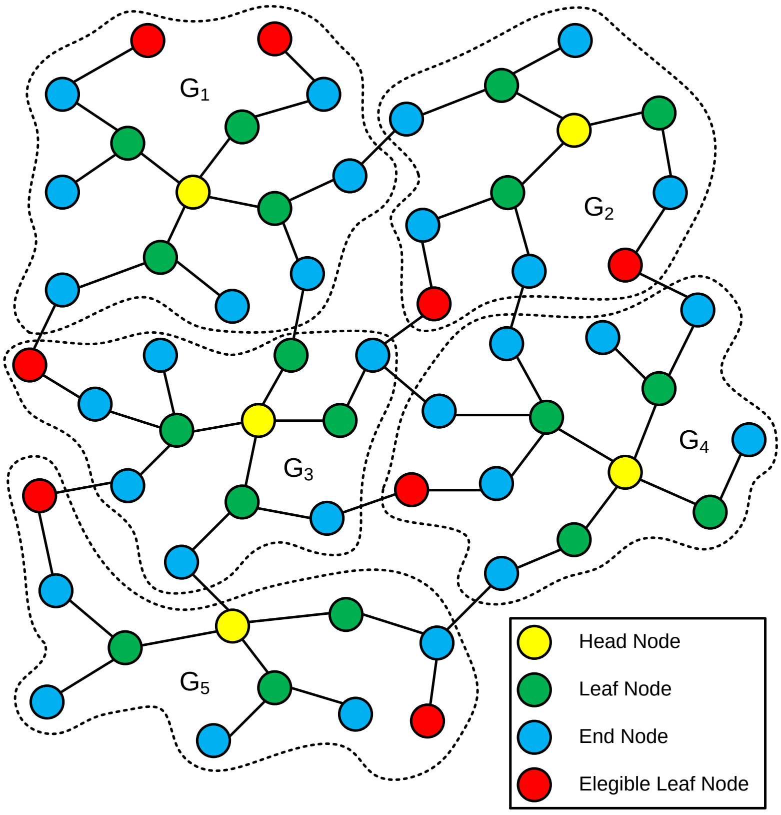


Figure 4-1: Proposed architecture topology.

When the information has to be sent to other groups, the information is routed directly to the closest end node to the destination group using the *groupID*. When a node from destination group receives the information, it routes it to all nodes in its group using OLSR protocol. Links between end nodes from different groups are established primarily as a function of their position, but in case of multiple possibilities, neighbors are selected as a function of their capacity. In order to establish the boundaries of the group, two choices can be considered: (i) limiting the diameter of the group to a maximum number of hops (e.g., 30 hops, as the maximum number of hops for a tracer of a route), and (ii) establishing the boundaries of the area that it is wanted to be covered.

When our group-based network is creating groups, a node may take different roles in different moments (they are analyzed in subsection 4.2.1). A node could start as a head node and when that group finish its topology it finishes as end node. We can see the relation between these roles in **Figure 4-2**. In this figure we can see that an ungrouped node can become any type of node depending on *HelloACK* messages. Then, these roles can change as long as our group topology varies. Also, if a node has leaf, end or eligible leaf role, it could become ungrouped node. This depends on group-creation rule.

As we will see in this chapter, our head node can go changing along group topology formation. So, the normal order to change a role of a node would be head node ←→ leaf node ←→ end node ←→eligible leaf ←→ ungrouped node.

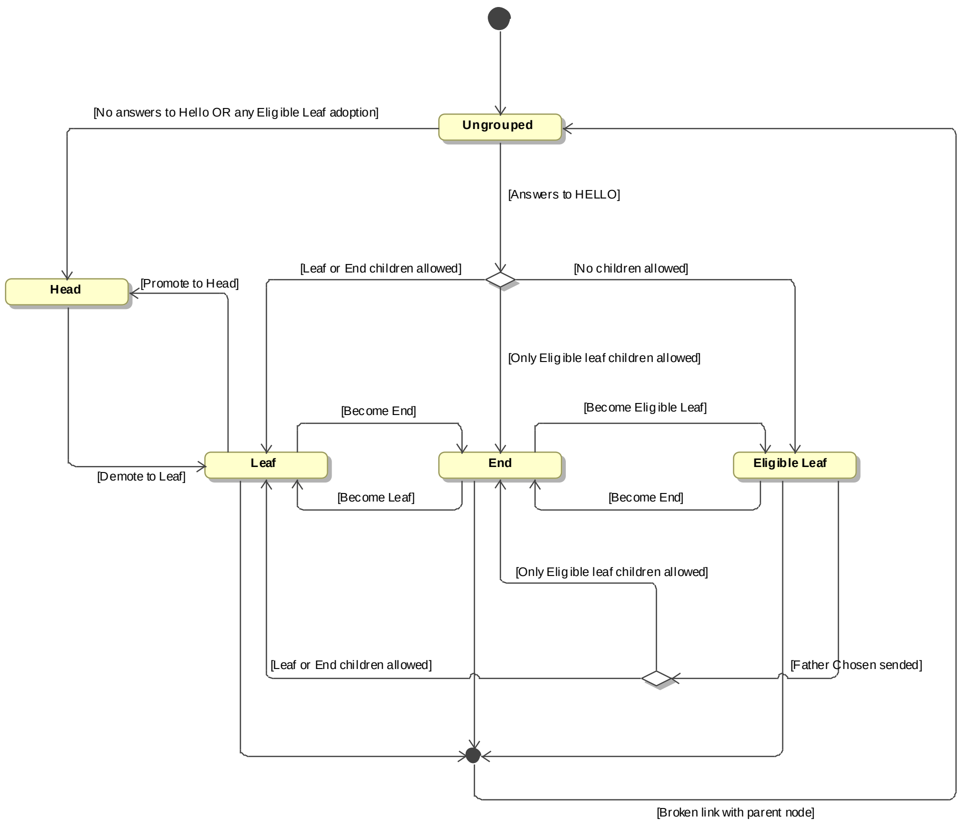

Ungrouped
Head
Leaf
End
Eligible Leaf
[No answers to Hello OR any Eligible Leaf adoption]
[Answers to HELLO]
[Leaf or End children allowed]
[No children allowed]
[Promote to Head]
[Only Eligible leaf children allowed]
[Become End]
[Become Eligible Leaf]
[Demote to Leaf]
[Become Leaf]
[Become End]
[Only Eligible leaf children allowed]
[Leaf or End children allowed]
[Father Chosen sended]
[Broken link with parent node]

Figure 4-2: Node's role state machine.

4.2.1 Node's Role

In this subsection, we are going to talk about node's roles. Each node may have a role in the process of creating groups. This role could change in life of a node, due to several things. For instance: network size, group size, number of neighbor nodes, etc.

In order to develop this architecture correctly, we have had to define five types of role for our sensor nodes. Next, these roles are defined:

- **Ungrouped Nodes**: A sensor node has this role when it is switched on, but it does not join to a group or it does not create a new group. When a node starts its activity, its first role will be ungrouped. This role goes on while it sends a *Hello* message and it has or not a reply. If it receives a positive replay, it will join to this created group. If it does not, it will create a new group. Main features of nodes with this role are: ungrouped nodes cannot have a father or sons, these nodes are not members of a group, these nodes sends *Hello* messages, *FatherChosen* messages, and they can receive *Hello* messages, *NewHead* messages. Other messages are not processed when a sensor node has this role. A sensor node can be able to this role when a node leaves a group and when it loses the connection with its father.

- **Head Nodes**: There will be only one node per group with this role (yellow nodes in **Figure 4-1**). The node with this role will be always on the center of physical/logical group's topology. In creation process of a group, our head node may be changing to be on center until our group achieves the rule, which defines maximum size of a group. When a node has this role its functions are the same than other nodes, but besides that, it is who control group creation. It is because the role of the rest of nodes in a group depends on the physical/logical position on the head. A node becomes a head node due to two reasons. The first one, it is because it is the first one in a group, in this case this node will create the *groupID*, too. And the second one because the size of a group is increasing and this can cause than our head node makes a head change, so when this event happens the groupID does not change. Main features of nodes with this role are: there will be only one head node per group, it controls the group's creation, it can make the groupID, it has sons, these nodes can sends *HelloACK* messages, *NewHead* messages, *HeadChange* messages, *KeepAlive* messages. They can receive *Hello*, *FatherChosen*, *DescendantLeft* and *KeepAliveACK* messages. Other messages are not processed when a sensor node has this role.
- **Leaf Nodes**: These nodes connect end nodes with head node of a group (green nodes in **Figure 4-1**). The number of leaf nodes depends on group's size, so if our group is large, we will have several leaf nodes. These nodes have two functions: a) sense environmental data and b) routing information throughout their group. This role can be changed while a group is deploying. It is due to that there could be head changes. This type of nodes appears in a group, while they are below the rule, which defines group's size. A node becomes a leaf node, when it sends a Hello message and it receives a reply of a Head node or a leaf node that is below the rule. Or in another case when there is a head change, because each head change could cause a role's change in the nodes of a group. Main features of this roles are: there could be several leaf nodes in a group, they have sons and a father (head or leaf node), these nodes can send *Hello, HelloACK, FatherChosen, NewSonInformation, KeepAliveACK* messages, they can relay *NewHead, HeadChange, AbandonGroup, DescendantLeft* messages, finally they can receive *Hello, FatherChosen, NewSonInformation, KeepAlive* messages. Nodes with this role can manage all messages (this messages will be defined in subsection 4.4).
- **End Nodes**: The main function of these nodes is to connect their group with other groups (blue nodes in **Figure 4-1**). Moreover, these nodes can connect leaf nodes or a head node (it depends on the group creation rule) with eligible leaf nodes (next we will see their definition). A node becomes

end node when it entries in a group and it just satisfies group creation rule. A node also becomes end node when there is a head change. This role and eligible leaf role are the nodes that allow exchanging information between groups. This feature helps us to improve our proposal in Chapter 6. Main features of this roles are: they allow connecting groups, there could be several end nodes in a group, they have sons and a father (head or leaf node), these nodes can send *Hello, HelloACK, FatherChosen, NewSonInformation, KeepAliveACK, DescendantLeft* messages, they can relay *NewHead, HeadChange, KeepAlive* messages, finally they can receive *Hello, FatherChosen* and *AbandonGroup* messages.

- **Eligible Leaf Nodes**: Finally, this kind of nodes is devised to have groups with more than one node (see red nodes in **Figure 4-1**). These nodes do not satisfy the group creation rule, so they should become a head node in a new group, but this drastic decision could create several groups with a node. For this reason we have selected that the first node, which does not satisfy the rule. It will be eligible leaf node and it will belong to current group. But if a new node appears, the eligible leaf node will leave its group as long as it improves its role. Their function will be the same as end nodes (they are explained before). Main features of this role are: they allow connecting groups, there could be several eligible nodes in a group, as long as they have a direct connection with an end node, they have a father (end node) and they do not have sons, these nodes can send *Hello, FatherChosen, AbandonGroup, KeepAliveACK* messages, finally they can receive *NewHead, HeadChange* and *KeepAlive* messages. Other messages are not processed when a sensor node has this role.

4.3 Group-based Architecture Analysis

4.3.1 Analytical Model and Neighbor Selection

Every node has 3 parameters (*nodeID*, C_{max} and λ) that characterize the node. C_{max} is maximum cost of each group, this cost is a metric for this reason it could be associated to several parameters as we will see in Eq. 4-2. Let λ parameter be the node capacity that depends on the node's upstream and downstream bandwidth (in Kbps), its number of available links (*Available_Con*) and its maximum number of links (*Max_Con*), its percentage of available load and its energy consumption. It is used to determine the best node to connect with. The higher λ parameter is the better node to connect it with. The λ equation is shown in Eq. 4-1.

$$\lambda = \frac{\left(BW_{up} + BW_{down}\right) \cdot Available_{Con} \cdot L + k_2}{Max_{con}} \cdot \sqrt{1 - \frac{E^2}{k_1}} \qquad \text{Eq. 4-1}$$

L is the available load and *E* is the energy consumption. Their values vary from 0 to 100. *E*=0 indicates it is fully charged, so λ parameter is 0 and *E*=100 indicates it is fully discharged. K_1 defines the minimum value of energy remaining in a node to be suitable for being selected as a neighbor. K_2 gives different λ values from 0 in case of L=0 or *Available_Con*=0. We have considered K_2=100 to get λ into desired values. **Figure 4-3** shows λ parameter values when the maximum numbers of links for a node are 16, for a bandwidth value of 128 kbps, as a function of its available number of links for different available energy values of the node. Node's load is fixed to 50%. **Figure 4-3** shows λ parameter values when the maximum numbers of links for a node are 16 and all have the same available number of links (*Available_con*=6) as a function of the node available energy for different bandwidth values. Node's load is fixed to 80%. It shows that as the Energy is being consumed, λ parameter is lower, but when it gets the 80% of consumption, the λ parameter decreases drastically, so the node is more likely to be chosen as a neighbor, in case of more available energy. **Figure 4-4** also shows that in case of a node with higher bandwidth is preferred.

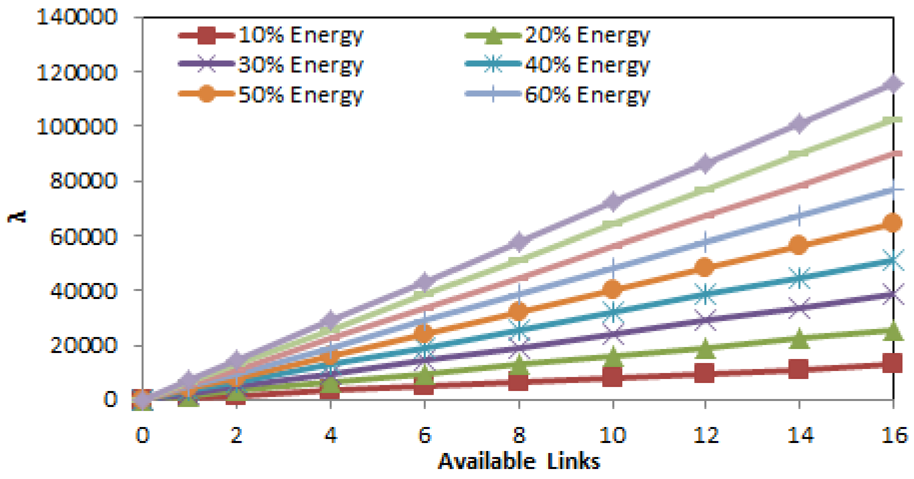


Figure 4-3: λ parameter values with number of links variation.

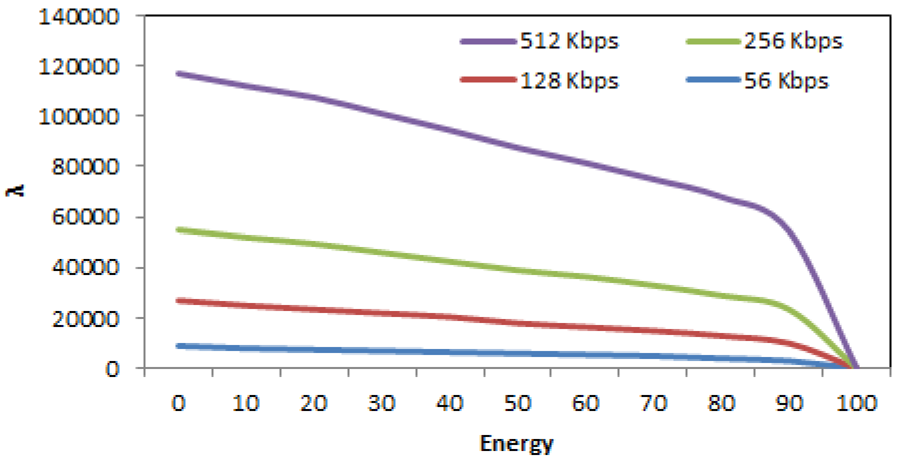


Figure 4-4: λ parameter values as a function of the energy of the node.

We have defined the cost of the ith node as the inverse of the ith-node λ parameter multiplied by T (the delay of its reply in ms). The cost is shown in Eq. 4-2.

$$C = \frac{T \cdot k_3}{\lambda} \qquad \text{Eq. 4-2}$$

K_3=10^3 gives $C \geq 1$. The metric for each route is based on the hops to a destination (r) and on the cost of the nodes (C_i) in the route as shown in Eq. 4-3.

$$metric = \sum_{i=1}^{r} C_i \qquad \text{Eq. 4-3}$$

The metric gives the best path to reach a node.

Let $G = (V, \lambda, E)$ be a network of nodes, where V is the set of nodes, λ is the set of their capacities ($\lambda(i)$ is the capacity of the i-th node and $\lambda(i) \neq 0 \ \forall$ i-th node) and E is the set of links between nodes. Let k be a finite number of disjoint subsets of V, so $V = \cup V_k$, and there is no node in two or more subsets ($\cap V_k = 0$), and let be $n=|V|$ (the number of nodes in V), the equation given for n is shown in Eq. 4-4.

$$n = \sum_{k=1}^{i} |V_k| \qquad \text{Eq. 4-4}$$

Every V_k has a head node, several leaf and end nodes and several eligible leaf nodes as shown in Eq. 4-5.

$$n = 1 + n_{leaf} + n_{end} + n_{eligibleLeaf} \qquad \text{Eq. 4-5}$$

Now we can describe the whole network as the sum of all these nodes from all groups as shown in Eq. 4-6.

$$n = \sum_{i=1}^{k} \left| \left(n_{head} + n_{leaf} + n_{end} + n_{eligibleLeaf} \right)_k \right| = k + \sum_{i=1}^{k} \left| \left(n_{leaf} \right)_k \right| + \sum_{i=1}^{k} \left| \left(n_{end} \right)_k \right| + \sum_{i=1}^{k} \left| \left(n_{eligibleLeaf} \right)_k \right| \qquad \text{Eq. 4-6}$$

On the other hand, the number of links in the whole network $m=|E|$ depends on the number of groups (k), on the number of links in each group (k_m) and on the number links between end nodes. Eq. 4-7 gives m value for a physical topology.

$$m = \sum_{i=1}^{k} \left(k_l + \frac{1}{2} k_b \right) \qquad \text{Eq. 4-7}$$

Where k_l is the number of links inside the group *k* and k_b is the number of external links of the group *k.*

4.3.2 Number of Nodes according to Group's Topology

In order to limit the coverage of the group, let's suppose there are a maximum number of hops between the head node and the most distant end node, using the shortest path. In this option, the geographic coverage of the nodes in a group will depend on the amount of nodes in some specific part of the group topology and on its position. The influence area is defined as all nodes near the head node, with an access time lower than T_{max}. It is supposed that the nodes are in a distance of d_{ij}. It can be observed in **Figure 4-5**, where T_{max} is specified according to the network and the distance d_{ij}, in number of hops. It will be the minimum distance between both nodes.

First, a maximum diameter of 30 hops has been taken. This indicates that there will have a maximum of 14 nodes between the head node and any end node. The maximum distance can vary depending on what is wanted: larger or smaller groups, but it will depend on the needs of the project in which the protocol is applied. When an event is noticed by a node of the group, it will be forwarded immediately to the entire group. So, a commitment between the number of nodes in the group and the number of groups needed to cover the entire node network, in order to minimize the convergence time, have to be achieved. The diameter depends on the topology of the nodes in the group.

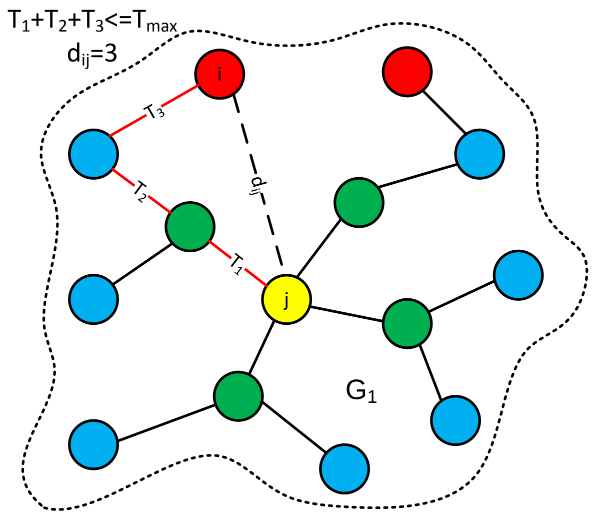


Figure 4-5: Head node influence area.

In a tree topology, the number of links is n-1, where n is the number of nodes in the tree. The diameter of the tree topology is shown in table III. M is the number of son nodes of a node in a tree topology (2 if binary, 3 ternary, etc.). The diameters of 2D and of 3D grid topology, with n nodes, are shown in table III. If the network scales freely, without control, following the Zipf law, when the maximum distance (dmax) have to be chosen, the cut-off effect shown by R. Cohen et al. in [89] [90] can be used. In networks where routing algorithms cross the network in few hops (such as Internet), d is defined as it is in table III. On the other hand, the logarithmic law states that the mean distance between two end nodes is equivalent to the diameter according to the law described in [91]. It can be calculated as it is in **Table 4-1**.

Table 4-1: Diameters of some topologies that could be used in groups networks.

Topology	***Diameter***
Tree	$d = 2 \cdot log_M[n \cdot (M-1)+1]-2$
2D Grid	$d = 2 \cdot \left(\sqrt[2]{n}-1\right)$
3D Grid	$d = 3 \cdot \left(\sqrt[3]{n}-1\right)$
Zipf law	$d \approx log(log(n))$
Logarithmic law	$ln \approx d \approx ln\left(\frac{n}{2}\right)$
P2P partially decentralized	$d = \frac{n}{16}$

Figure 4-6 shows the comparative between the studied topologies. The lower diameter is Zipf law with a R coefficient of -2,45. Nevertheless, when there are less than 14 nodes in the network, the worst case is the binary tree topology. Between 15 and 22 nodes, both binary tree and 2D grid topologies are the worst ones, and when there are more than 23 nodes, the worst case is the 2D grid topology.

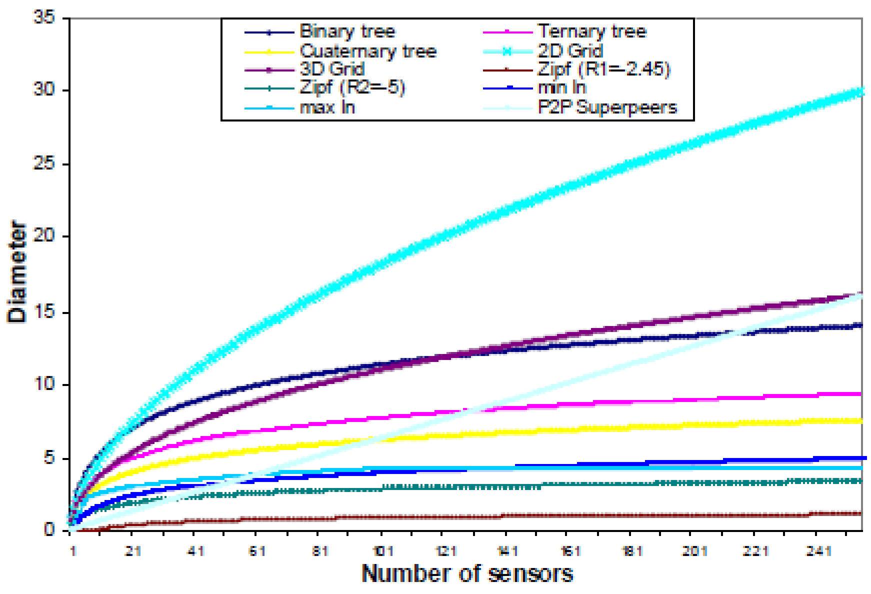


Figure 4-6: Diameter according to the total number of nodes.

4.4 Messages

In this subsection we are going to show messages designed. Several fields for future purposes, such as version, length and options fields, have been included. In order to reserve enough bits for the parameters 16 bits have been used for the λ, the GroupID and the nodeID parameter. These messages have to work together with the routing protocol used in the group.

4.4.1 Hello Message

This message is sent when an ungrouped node wants to become part of a group or to create a new group. This message begins activity in the group-based network. When a node sends this message, it expects to receive a HelloACK message, but this could not happen. For this reason, there is a timeout process, which is started when a Hello message is sent. If this timeout finishes without any reply, this ungrouped node will become Head. There are 6 fields used in this message (see **Figure 4-7**). The first one is “Version” that indicates the version of our protocol, in our case 1. The second one is “Length” who defines the length of this packet header, in this case 70 bits. Then, we have “CommandID”, it assigns a number for this message, Hello message = 1. “SrcID” and “DestID” are identifiers of source node and destination node, if receiver nodes are all nodes in a group we use 255. Finally, we will have “SeqNumber” to reply to a node correctly.

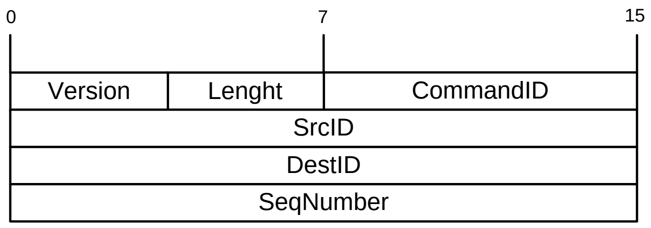


Figure 4-7: Hello message.

4.4.2 HelloACK Message

HelloACK message is the replay of a Hello message. If a node sent a Hello message with a HelloACK as a replay, the same node would know which are its neighbors, which will be its GroupIDs, its HeadIDs and the distance until its head nodes. A node can receive several HelloACK from several nodes, but it will only choose one. This chosen node will have the highest λ parameter. The first 6 fields of the HelloACK message are the same than the first 6 fields of Hello message. Then, we have “GroupID”, it identifies the ID of the group where node goes to join. “HeadID” is the ID of the head of that group. Next, we have “HeadDistance”, which show the distance between the node that sends this message and its head node. We have “λ”, this parameter is used to select the best neighbor, as we have seen in the next section. Finally, “NodeType” field, which identifies the node role.

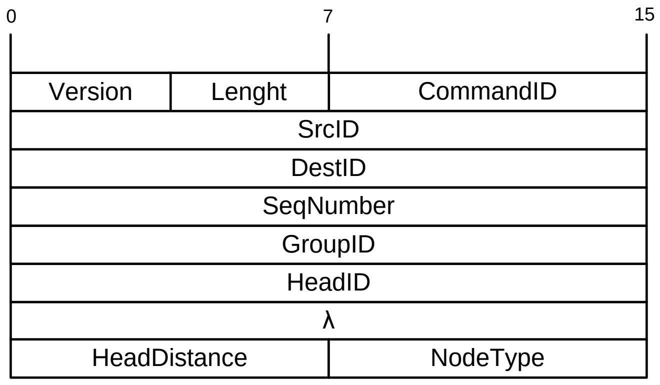


Figure 4-8: HelloACK message.

4.4.3 NewHead Message

This message can be only sent by a head node. With this message a head node informs that a new group has been created. In this case, the DestID is equal than 255. GroupID = new group ID and HeadID = ID of node, which sends this message. In this case HeadDistance = 0, this field could be removed but we have thought that if we include this field in this message, the message's creation will be more stable.

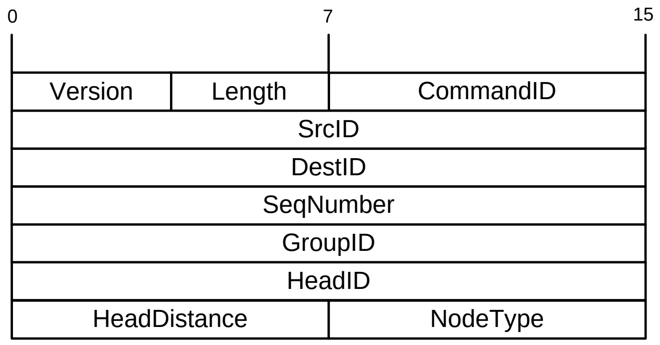


Figure 4-9: NewHead message.

4.4.4 FatherChosen Message

FatherChosen message is used to notify that a node has selected by another one as neighbor. This message will be sent in two times. One, when a node had received several HelloACKs and it had evaluated them and had selected the best neighbor. It will be send a FatherChosen message to its father. Another situation could be when an eligible leaf listen a HewHead message. As it is known an eligible leaf does not satisfy the group-creation rule, so it can change its role for another better, it will do without any problem. A sender node will take part in a new group with this message. In this case "CommandID" = 4, "DestID" = father's ID, "GroupID" = father's group ID, "HeadID" = father's head ID, "HeadDistance" = father's head distance + 1, and finally the node type that it will be.

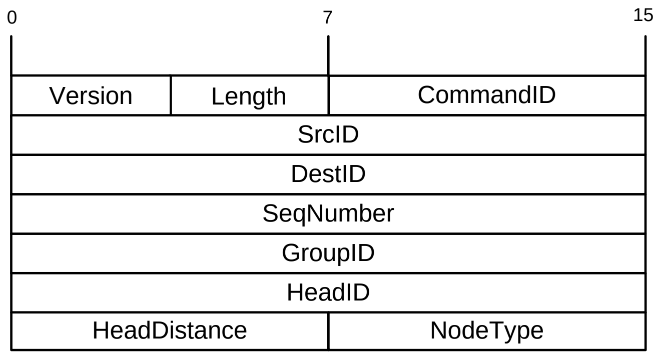


Figure 4-10: FatherChosen message.

4.4.5 NewSonInformation Message

This message is sent when a node receives a FatherChosen, except for a head node that this information is known with a FatherChosen message and an eligible leaf, which cannot be able to have son nodes. This message helps to a head node to create its group correctly. This message will communicate the data of new sons to their fathers. These messages are sent until head node. Its fields are the same than the fields of a FatherChosen message, but in this case there is a new field called ExtraData1. In this field will be the information needed to do a head change when the network topology is not centered. In this case "CommandID" = 5, "DestID" = father's ID, "GroupID" = my group ID, "HeadID" = my head ID, "HeadDistance" = my head distance, the node type that I am, and finally the data to make a head change.

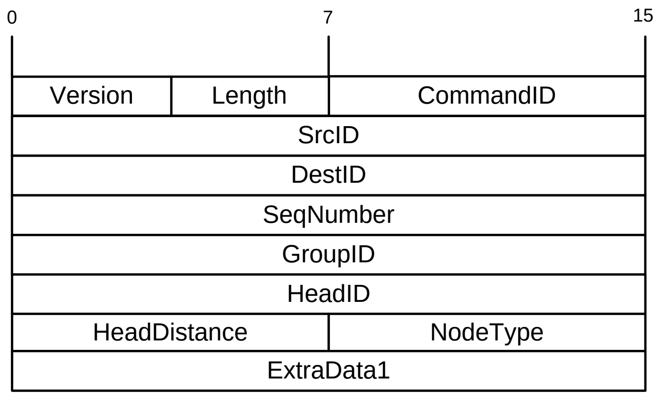


Figure 4-11: NewSonInformation message.

4.4.6 HeadChange Message

The creator of this message is the current head node. According to information sent by the FatherChosen message, the head node is seeing if the network topology is centered. When it detects that a neighbor can be a most central node sends a message HeadChange. This message is sent to the neighbors and they spread it to the whole group knows the change. The fields in this message will possess the information of the new head. In this case the GroupID is unchanged, but the HeadID changes. In the first message, the headDistance will be updated with the new value, like the new node type (different from head node role). ExtraData1 and ExtraData2 fields will be required to update the topology data.

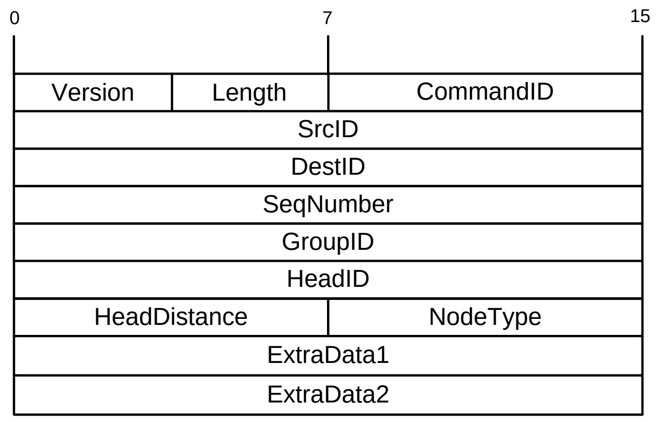


Figure 4-12: HeadChange message.

4.4.7 AbandonGroup Message

An AbandonGroup message is sent by a node when it wants to join to another group. This message will be sent primarily by eligible leaf nodes, because they do not satisfy the rule of group creation. When these nodes listen a FatherChosen or NewHead message, which would assess the new situation. If this improves the current situation will leave the group to join the new group. Before a node leaves its group, it sends an AbandonGroup message to inform the group that this node will not belong to that group. This message will be sent to the parent node of the node that leaves the group. Then the Father will send updated information to the head node with another message. The fields used in the observer can.

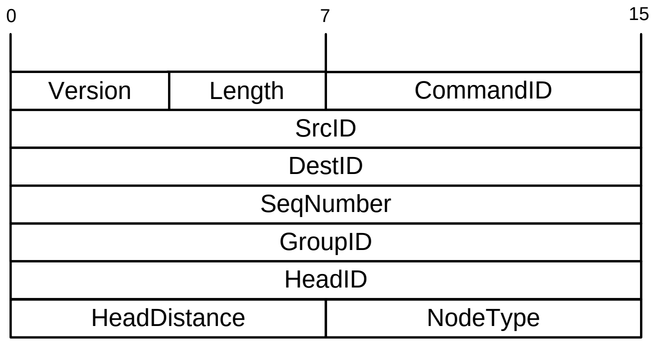


Figure 4-13: AbandonGroup message.

4.4.8 DescendentLeft Message

The DescendantLeft message is used to update all the topology of the group when an AbandonGroup. This message is addressed to a parent node and this in turn sends it to its parent node haste to get to the head node. The information included in this message can observe in Fig. As we can use the field observer ExtraData1 where metric data traveling realizer enable updating the cluster architecture.

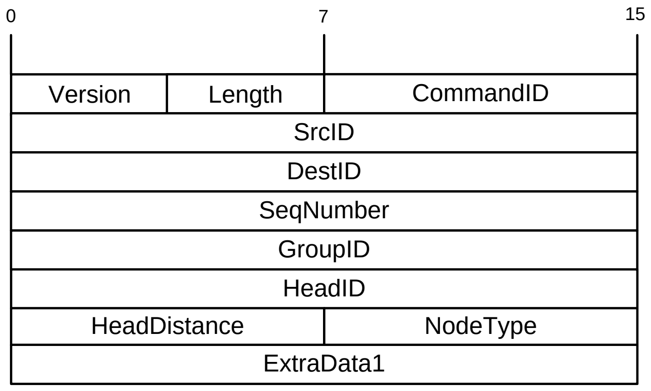


Figure 4-14: DescendentLeft message.

4.4.9 KeepAlive Message

Through this message we can see that the nodes of our group are still active. This message may be sent by any node, but normally shaped realizer keepalive task will be the following parent nodes send the message to their children and they respond with the same message ACK bit set and adding information to your state ExtraData1 field, which in this case have 15 bits. This information will be information related to the node, such as battery level, if this level is very low node parent knows that will die, so in a shorter time interval to the interval of keepalive send another message if cnt is because he died and therefore the father sent a message of type DescendantLeft to update the network topology.

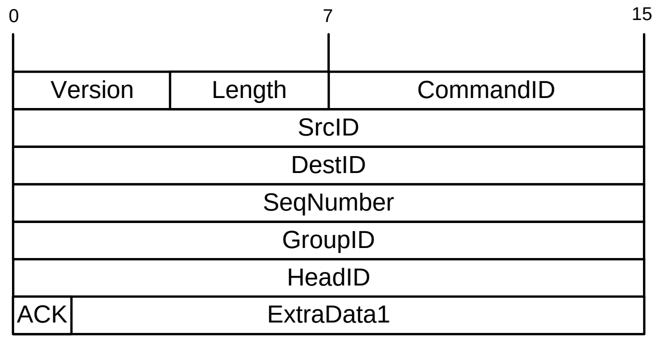


Figure 4-15: KeepAlive message.

4.5 Group-based Protocol Operation

This section describes how designed protocol operates. This section is divided into three subsections.

4.5.1 Group Creation and Maintenance

Let a new node join the network (it could be the first). It sends a Hello message in order to join a group. If there is no response from any node for a time, the node considers itself as a head node of a group in the network, and it will take the value *groupID*=1 and *nodeID*=1.

When the node receives *HelloACK* messages from several candidate neighbors, first it puts a timestamp on their reply and chooses the best nodes to have a link with (this election is taken based on the λ parameter which comes in the *HelloACK* message). The timestamp and head distance will be used to calculate *C* parameter. Responses received after timeout will be discarded. In case of receiving replies from nodes of different groups, it will choose the group whose replies have the highest average λ parameter on condition that they satisfy the rule of group creation (for i^{th} node $C_i <= C_{max}$), so it will take into account replies only for that group.

Each node has a parameter known as C_{max}, a node can belong to that group as its C <= C_{max}. In one exceptional case C can outperform C_{max}. In this case our node would become eligible leaf node.

Then, the node will send a FatherChosen message to its father. This message will have every parameter with their correct information to join a new group. When this father node receives this message, it will know that a new node has been added to its group. So, if it is a node with a different role than a head node, it will send a NewSonInformation message to update the information of that group. This process could be seen in **Figure 4-16**. The link will be established when a node receives a FatherChossen message from node, which starts sending a Hello message.

Nodes will send *KeepAlive* messages periodically to their neighbors. If a node does not receive a reply of a *KeepAlive* message from its neighbor before the dead time, it will remove this entry from its database and will start the group update process using the DescendantLeft message. As the *groupID* does not change in a group, the updating process is easy until failed nodes are the last nodes of a group. When a father node fails, it might happen two situations:

- Some son's nodes do not have descendants, so in this case they will become ungrouped nodes. When they are ungrouped nodes, they will start the creation process again sending a Hello message.
- Some son's nodes have descendants. In this situation the father of these descendants will become head node. These new head nodes will send a NewHead message with a new groupID. It will create a new group. This message will be processed by all nodes in order to update your group information.

Links between end or eligible leaf nodes from different groups are established as a function of their *C* metric of the replying nodes, but it could be changed by an algorithm using node's position or choosing the neighbor with the lowest distance (in number of hops) to the head node. If we base our proposal on the λ parameter, we will distribute the load of the network between groups, but if we base our proposal on a node's position or choose the neighbor with the lowest distance to the head node we will balance the number of nodes in the groups.

When a new node joins the group, the head node of the group could be changed because in our proposal the head node should be as centered as possible. The procedure designed for changing the head node is explained as follows.

While a group is growing, there will be several head changes in order to center this node in the group-based topology. For this reason when there is a change of head, head node sends a HeadChange message to its sons. This message will have the updated information about the new head. At the same time its sons also send this information until the last nodes of the group. Head node knows how many descendants have its sons and their weight (see an example in **Figure 4-17**). With values a head node evaluates if it needs to move to one direction or to another. This information is updated when each node receives a NewSonInformation message.

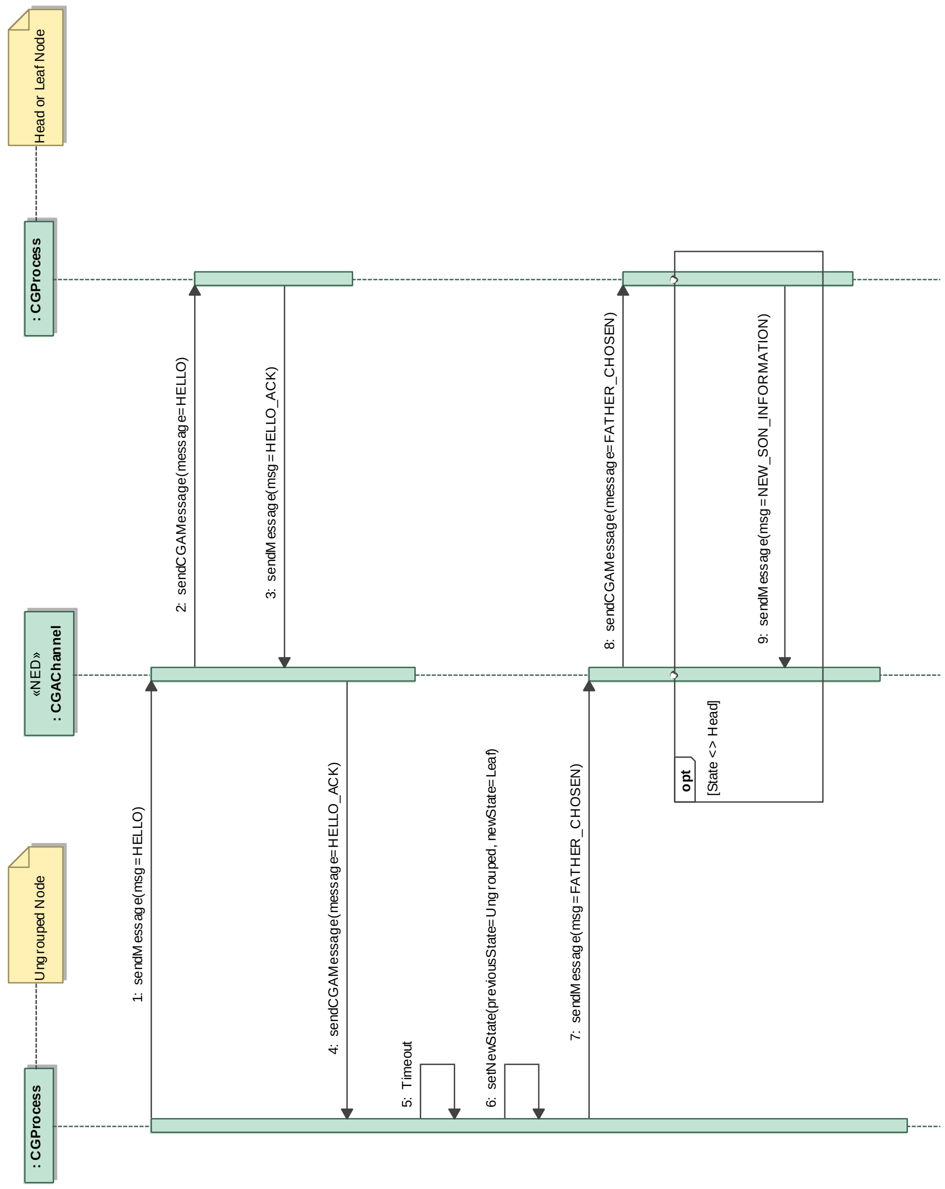

Head or Leaf Node
: CGProcess
«NED»
: CGAChannel
Ungrouped Node
: CGProcess
1: sendMessage(msg=HELLO)
2: sendCGAMessage(message=HELLO)
3: sendMessage(msg=HELLO_ACK)
4: sendCGAMessage(message=HELLO_ACK)
5: Timeout
6: setNewState(previousState=Ungrouped, newState=Leaf)
7: sendMessage(msg=FATHER_CHOSEN)
8: sendCGAMessage(message=FATHER_CHOSEN)
9: sendMessage(msg=NEW_SON_INFORMATION)
opt
[State <> Head]

Figure 4-16: Group creation process.

In **Figure 4-17** we can that head node has three branches. In the left side, it has 4 descendants with a weight of 8. Sons have a weight equal than 1, the sons of this sons their weight is 2 per branch and last nodes has a weight of 3. As we can see this weight is related to distance between a node and the head node of a group.

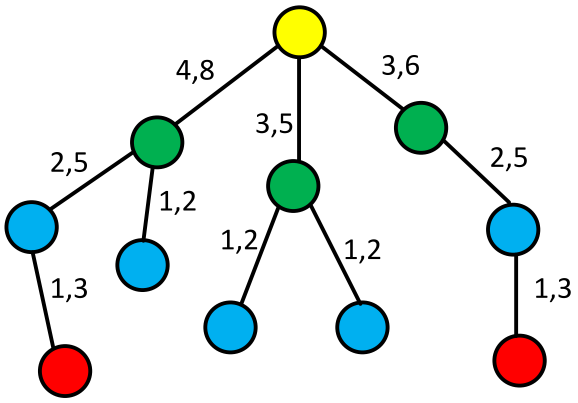


Figure 4-17: Group-based logical topology with information (descendants, weight).

Steps followed by a head change can be seen in **Figure 4-18**. According to new nodes join to a group, they send NewSonInformation messages. These messages are processed and the head node starts a process called HeadChangeEvaluator. This evaluator can take into account only number of descendants or number of descendants and weight. This process informs to head node what is the best node to be head. When this head node knows this information, it sends the message HeadChange with the data of the new head node.

When there is a change of the head node of a group, all the nodes in the group must be alerted. In order to update all nodes in the group, the previous or the new head node will send a *HeadChange* message to indicate the new head node and the updated data from it to the node processing this control packet. This update is distributed using the Recursive Proportional-Feedback (RPF) algorithm.

Once the links between neighbors are established, every node sends *keepalive* messages periodically to its neighbors.

We define the group diameter (d_{group}) as the shortest number of hops, between the two most remote nodes in the group (in our case, $d_{group} \leq 30$). We have proposed two choices to establish the boundaries of the group:

1. When the boundaries of the group are the same as the area that is to be covered, end nodes are known using GPS.
2. When the boundary of the group is limited by the diameter of the group, the maximum number of hops from the head node must be known. Every time a new node joins a group, it receives the newNode ACK message with the number of hops to the head node. When it achieves the maximum number of hops, the node is marked as a end node, and it will inform new joining nodes that they must create a new group.

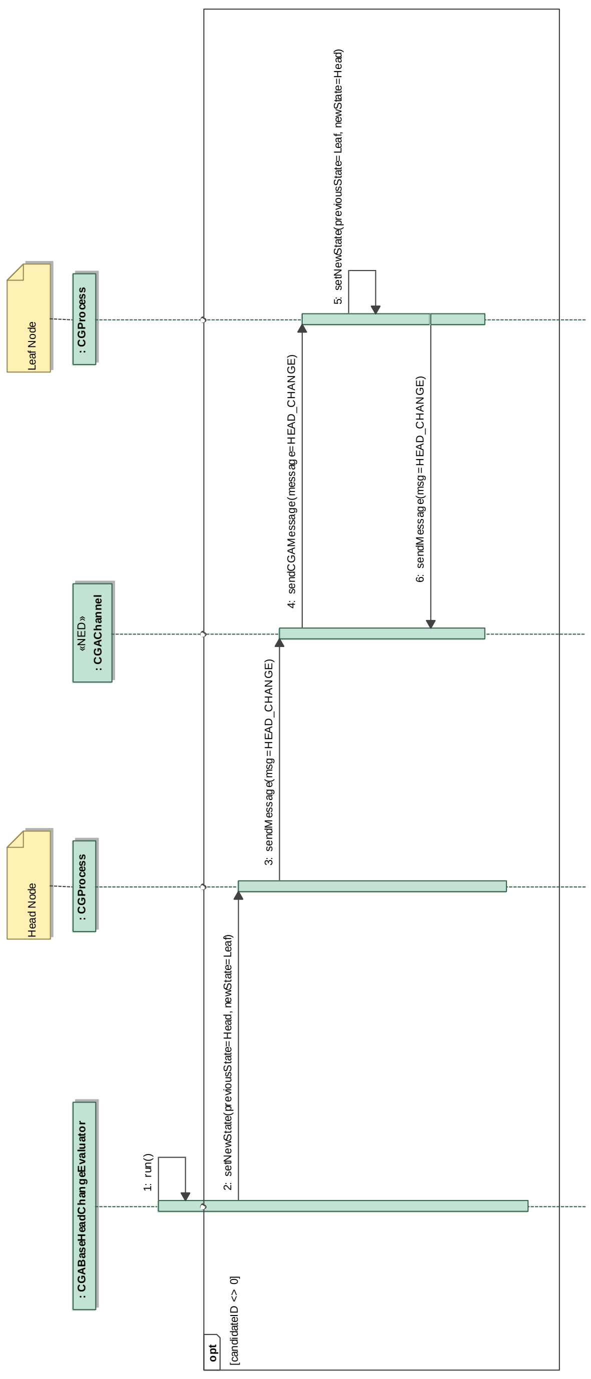

Leaf Node
: CGProcess
«NED»
: CGAChannel
Head Node
: CGProcess
: CGABaseHeadChangeEvaluator
opt
[candidateID <> 0]
1: run()
2: setNewState(previousState=Head, newState=Leaf)
3: sendMessage(msg=HEAD_CHANGE)
4: sendCGAMessage(message=HEAD_CHANGE)
5: setNewState(previousState=Leaf, newState=Head)
6: sendMessage(msg=HEAD_CHANGE)

Figure 4-18: Head change process.

4.5.2 Leavings and Fault Tolerance

When a node leaves the group, it will send AbandonGroup message to its neighbor nodes. They must send a DescendantLeft message to their fathers in order to update the information of this group. This information is important to know if our head node is on center in its group. This DescendantLeft message is resent to the head node. DescendantLeft message and NewSonInformation are two important messages in order to have or group topology as centered as possible, without them our topology does not know if our head node is centered or not.

A DescendantLeft message can bring out a head change. When this message arrives at the head node, it runs its HeadChangeEvaluator and it will decide if its group topology needs a head change. If this decision was positive, it would send a HeadChange message.

In **Figure 4-19** we have the most likely situation of a group leaving. In this case we have a group with an eligible leaf node. A new node switches on and its first state is ungrouped. It sends a Hello message to join a group or to create a new group if it does not receive any reply. In this case the Hello arrives to the eligible leaf node, but it does not reply. For this reason, the ungrouped node becomes head node and it sends a NewHead. This last message is listened by eligible leaf node. When it receives a NewHead message it means that it has a neighbor node that gives itself a good role. So, it will send an AbandonGroup to its current group and it leaves this group. Finally, it will send a FatherChosen to the new head node in order to join that new group created by the ungrouped node.

If the leaving node is the head node, it assigns head node role to the best candidate to be the headnode (in case of draw, it will choose the one with higher λ parameter, this node is called backup head node and will be always kept advised by the head node) using the HeadChange message. Then, the new head node will resend this message to the group to inform the change. If the head node fails down without any advertisement, this situation is explained in the subsection 4.5.1. This procedure would be done in the same way with a node leaves its group with an AbandonGroup message. With message the group should not wait until a KeepAlive arrives. So, the procedure would be quitter than the previous one.

When a regular node fails down, its neighbor nodes will know the failure because of the missing of KeepAlive messages. The procedure is the same as when the node leaves the network voluntarily. Neighbor nodes will delete that entry from their database and if a node keeps without any neighbor, it will begin the discovery process as a new node in the network (see subsection 4.5.1).

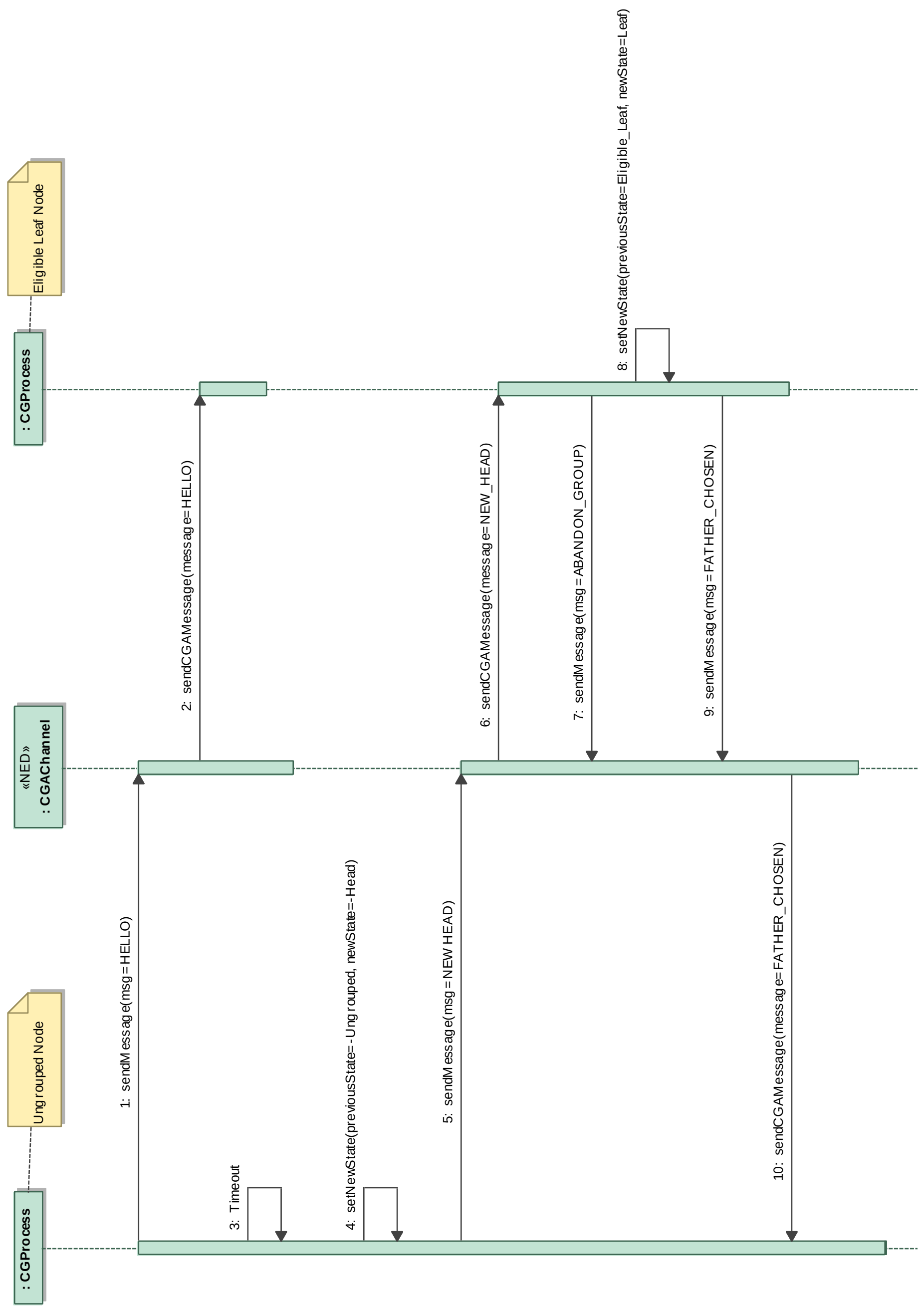

Eligible Leaf Node
: CGProcess
«NED»
: CGAChannel
Ungrouped Node
: CGProcess
1: sendMessage(msg=HELLO)
2: sendCGAMessage(message=HELLO)
3: Timeout
4: setNewState(previousState=-Ungrouped, newState=-Head)
5: sendMessage(msg=NEW HEAD)
6: sendCGAMessage(message=NEW_HEAD)
7: sendMessage(msg=ABANDON_GROUP)
8: setNewState(previousState=Eligible_Leaf, newState=Leaf)
9: sendMessage(msg=FATHER_CHOSEN)
10: sendCGAMessage(message=FATHER_CHOSEN)

Figure 4-19: Abandon group process (eligible leaf node case).

4.5.3 Connection between Groups

Once a node has joined a group, it is able to send and receive KeepAlive messages as it has been explained before. Nodes have 2 tables, first one has all its neighbors of its group and the second one has all its neighbors from other groups. When a node receives a broadcast KeepAlive message from a neighbor node that belongs from another group, it checks if it is in its other groups' neighbors table. If it is inside, it resets the KeepAlive time for that entry to zero, but if it is not inside, it adds that entry to the table. The flow chart shown in **Figure 4-20** explains this procedure.

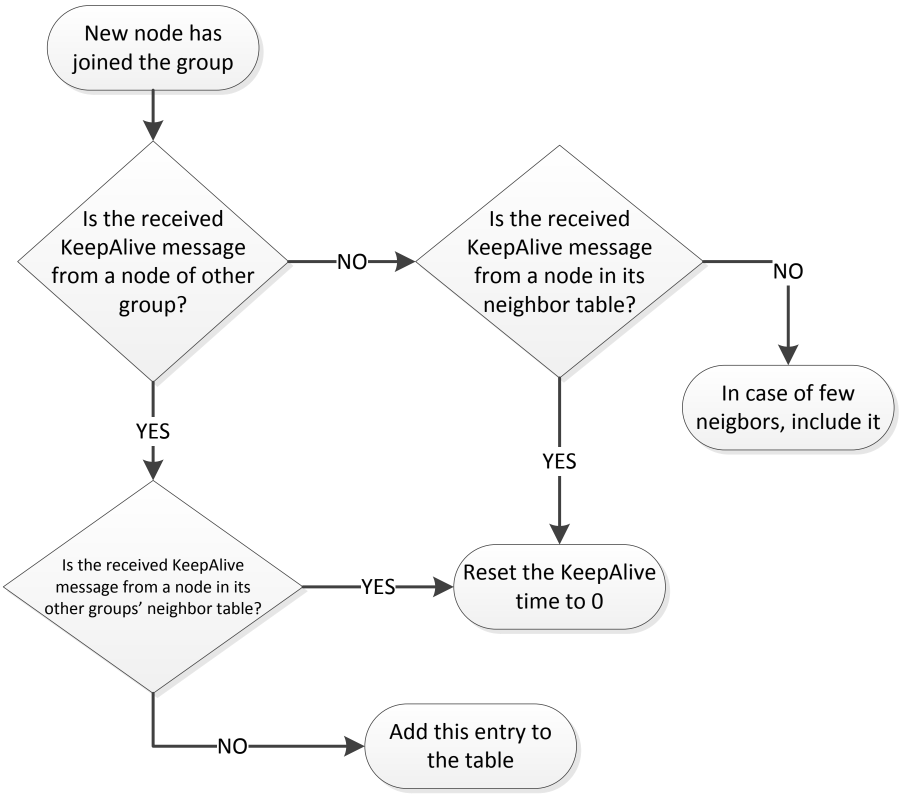

New node has joined the group
Is the received KeepAlive message from a node of other group?
NO
Is the received KeepAlive message from a node in its neighbor table?
NO
In case of few neigbors, include it
YES
YES
Is the received KeepAlive message from a node in its other groups' neighbor table?
YES
Reset the KeepAlive time to 0
NO
Add this entry to the table

Figure 4-20: Flowchart to establish links with nodes from other groups.

4.6 Connection Time Simulations

Let Ti be as the time needed by two nodes to communicate with each other, and RTT (Round Trip Time) as the mean value of the round trip time between both nodes. So, Ti can be calculated using Eq. 4-8.

$$T_i = \frac{RTT_i}{2} \qquad \text{Eq. 4-8}$$

The time needed to communicate a source node with a destination node in a different group is calculated using the expression given for Tmax_intergroup in Eq. 4-9. In this study we assume that our groups are perfectly created, for this reason there are not eligible leaf in or groups. Each group will have a head node and several leaf and end nodes.

$$T_{\max\_intergroup} = t_{source\_end} + \sum_{i=1}^{n} t_{\max\_intragroup\_i} + \sum_{i=1}^{n+1} t_{end_{i}\_end_{i+1}} \qquad \text{Eq. 4-9}$$

n is the number of intermediate groups, t_{source_end} is the time needed to arrive from the source node to the end node in the same group, $t_{max\_intragroup\_i}$ is the time required to go through the i-th group, and $t_{end\_i\text{-}end\_i+1}$ is the time needed to transmit the information from the end node of a group to the end node of another group connected to the previous one.

We define t_p as the average propagation time for all the message transmissions between two nodes in the architecture. Its expression is shown in Eq. 4-10.

$$t_p = \frac{\sum_{i=1}^{m} T_i}{m} \qquad \text{Eq. 4-10}$$

m represents the number of nodes involved in the path minus one. Taking into account t_p, the time needed to transmit information from the source node to the end node of the same group (T_{source_end}) is defined in Eq. 4-11.

$$T_{source\_end} = d_{source\_end} \cdot t_p \qquad \text{Eq. 4-11}$$

d_{source_end} are the number of hops needed to arrive form the source node to the end node of the same group. The maximum time to cross through a group ($T_{max\_intragroup\_i}$) is defined by the expression shown in Eq. 4-12.

$$T_{\max\_intragroup} = d_i \cdot t_p \qquad \text{Eq. 4-12}$$

i indicates the group and the d_i is the number of hops in the group. On the other hand, the number of hops for *j* groups is shown in equation 13.

$$d_i = \sum_{j=1}^{j} d_j \qquad \text{Eq. 4-13}$$

Replacing in Eq. 4-10, Eq. 4-11, Eq. 4-12 and Eq. 4-13 in Eq. 4-9, we obtain Eq. 4-14.

$$T_i = \left(d_{source\_end} + \sum_{i=1}^{n} d_i + n + 1 \right) \cdot t_p \qquad \text{Eq. 4-14}$$

In figure 31, we see how the interconnection time evolves between nodes of different groups. In the following subsections we are going to use Eq. 4-14 in order to model our proposal.

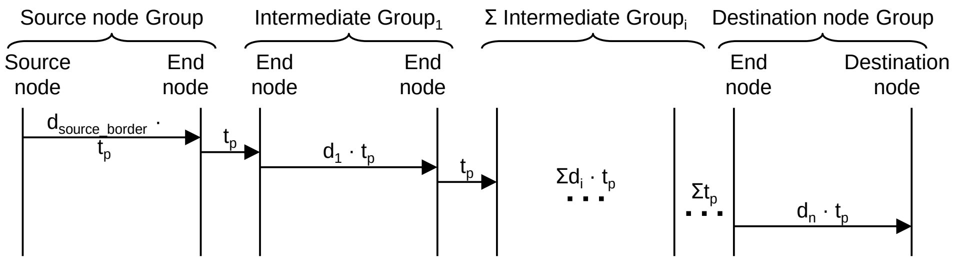


Figure 4-21: Connection time between nodes of different groups.

4.6.1 Connection Time Variation when Groups have Same Number of Hops

This simulation is done fixing the number of intermediate groups and varying the number of hops between source node and the end node of its group. Then, we can observe what happens when the number of hops of the intermediate groups increases.

We have chosen the number of intermediate groups as 4. Considering that all the intermediate groups have the same number of hops, it means $d_1=d_2=d_3=d_4=d$, and introducing these values in Eq. 4-6 we obtain the Eq. 4-15.

$$T_{\max\_intergroup} = \left(d_{source\_end} + 4 \cdot d + 5 \right) \cdot t_p \qquad \text{Eq. 4-15}$$

When we give higher values to d_{source_end} for each value of d, the maximum inter group time ($T_{max\_intergroup}$) increases lineally.

4.6.2 Connection Time Variation when Number of Hops to Cross Groups Varies

This sub-section studies what happens when we maintain the distance between source node and the end node of the source group constant and we vary the number of hops of the intermediate groups and for different number of groups. We fix the parameter d_{source_end} to a value of 10. Using Eq. 4-14, Eq. 4-16 is obtained.

$$T_{\max\_intergroup} = \left(11 + \sum_{i=1}^{n} d_i + n \right) \cdot t_p \qquad \text{Eq. 4-16}$$

Now, we can vary d_i to observe the time needed to achieve its destination. Results are shown in **Figure 4-22**. We can deduce that the number of groups in a network does not affect the connection time to a large extent when the mean number of hops to go through the groups is low. Nevertheless, when the mean diameter of the groups is big, increasing the number of intermediate groups implies a large increase in the connection time. So, we can state that the mean diameter of the groups becomes more relevant in the calculation of the final connection time ($T_{max\_intergroup}$) for bigger networks.

In **Figure 4-23**, we can observe how the connection time varies according to the number of groups for different numbers of hops. We have chosen d_{source_end} = 20, and we have varied the number of groups that will be crossed for different mean diameters of the groups, instead of varying the mean diameter of the groups.

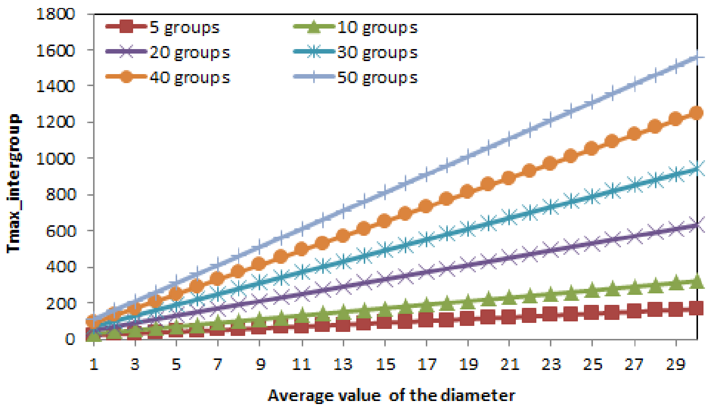


Figure 4-22: $T_{max\_intergroup}$ variation according to the mean diameter of group.

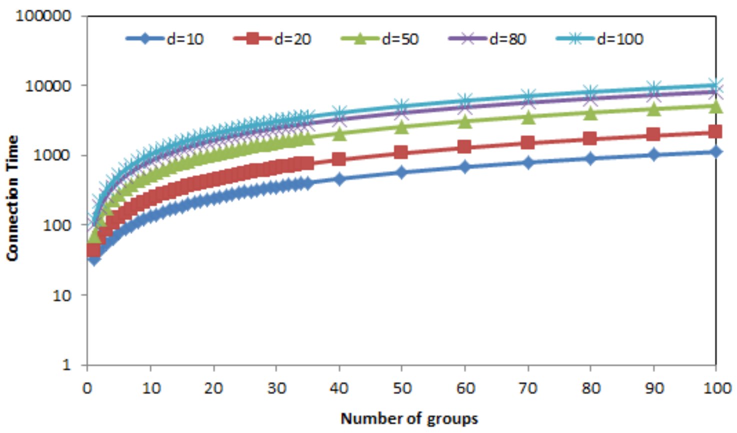


Figure 4-23: Connection time variation for different diameters.

4.6.3 Connection Time Variation for Different Number of Groups and Different Distances between Source and End Nodes in a Group

In this section we analyze how the maximum inter group time varies when we maintain the mean diameter of the group as a constant value and vary the number of groups for different distances between source and end nodes of the same group. To perform this experiment, we have chosen 20 as the mean diameter of the groups. Eq. 4-17 shows the connection time depends on the distance between the source and end nodes in the same group and on the amount of groups in the network.

$$T_{\max\_intergroup} = \left(d_{source\_end} + 21 \cdot n + 1\right) \cdot t_p \qquad \text{Eq. 4-17}$$

Figure 4-24 shows the behavior of the $T_{max\_intergroup}$ as a function of n for several d_{source_end} values. The maximum inter group time (t_p) increases when the number of intermediate groups increases. This has happened in all the analyzed cases. Nevertheless, as we can see, there is not a big difference in the final time when we have a large or short distance between source and end node (d_{source_end}). It means the number of hops is more relevant than the number of groups than d_{source_end} for having better $T_{max\_intergroup}$. This is an important subject to take into account when designing node networks.

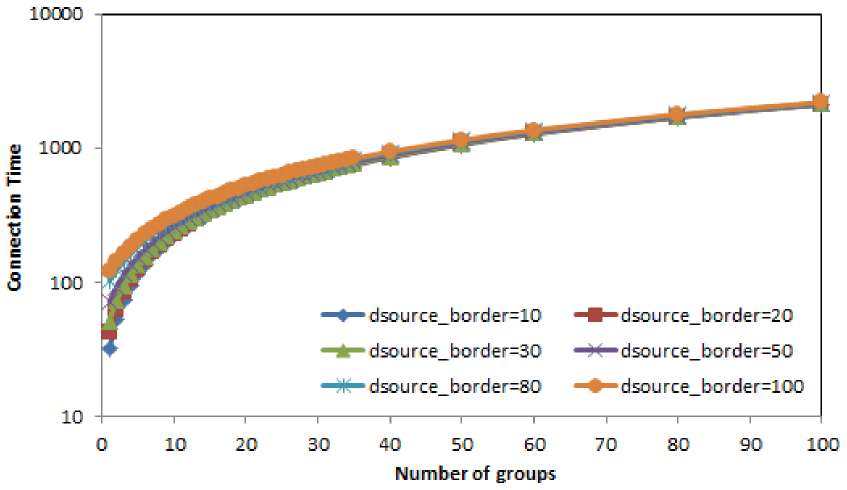


Figure 4-24: Connection time variation according to the number of hops to end nodes.

4.6.4 Connection Time for Getting a Destination Group according to Groups' Diameter

In this sub-section, we show the results of several simulations that give us an objective point of view about how to design a group based node network to obtain a short connection time between two nodes belonging to different groups.

We have simulated the time needed by a message sent by a node in a group until it arrives at an-other node of another group. Then, we observed the variation of the number of hops and the variation of time needed to reach the destination group.

In **Figure 4-25** we see the connection time of two groups in a network with 4 groups. In order to obtain the series of the source group, we have fixed a value of 10 hops for the mean diameter of the groups and the diameter of the source group has varied between 1 and 30 hops (we have considered that groups have a maximum diameter of 30 hops). To obtain the series of the mean diameter of the group, we have fixed a value of 10 hops for the diameter of the source group and the mean diameter of the groups varies between 1 and 30 hops. As we can see, the connection time between 2 nodes increases more when the mean diameter of the intermediate groups increases. Moreover, the interconnection time between two nodes is not so significant when the diameter of the source group increases.

In **Figure 4-26**, we have simulated the connec-tion time between two nodes of a network with 20 groups. In order to obtain the series of the source group, a mean diameter of the groups of 20 hops has been fixed and the diameter of the source group has been varied between 1 and 30 hops. To obtain the series of the mean diameter of the group, a diameter of the source group of 20 hops has been fixed and the mean diameter of the intermediate groups varies between 1 and 30 hops.

In **Figure 4-25** and **Figure 4-26**, we can observe that the delay (connection time) increases when the mean diameter of the groups increases, but that increase is less significant when the number of hops from the source node to the end node of the same group increases, as we expected. Note that when we want to design a group-based net-work with many groups, the best solution is to increase the mean diameter of the intermediate groups instead of increasing the diameter of the source group. When a network with few groups is needed, the interconnection time varies less when we increase the number of hops in the source group.

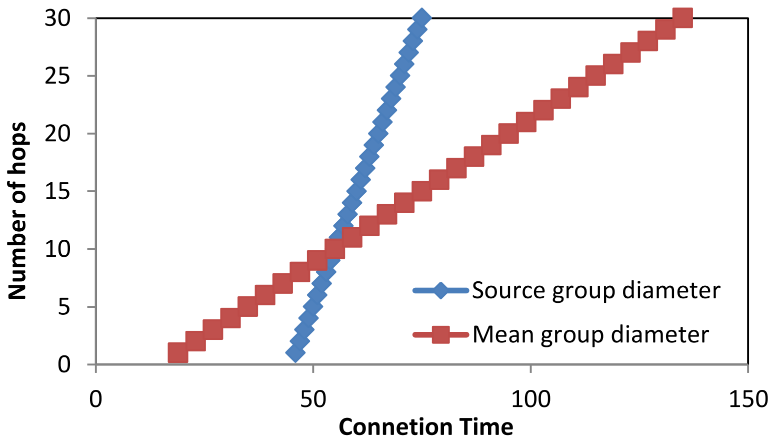


Figure 4-25: Connection time to reach the destination group according to the number of hop (4 groups).

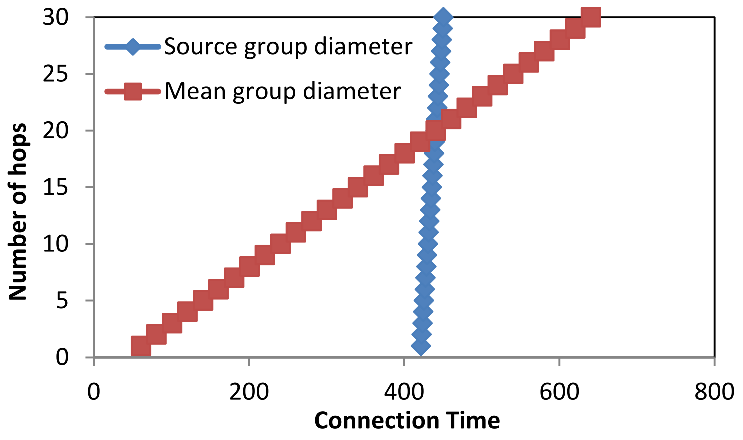


Figure 4-26: Connection time to reach the destination group according to the number of hop (20 groups).

4.7 Network Comparison

This section shows the comparison of our proposal with other planar group-based networks. The first one is the proposal of Xiang et al. (a locality-aware overlay network based on groups [31] is proposed, which has been used for Peer-to-Peer Based Multimedia Distribution Service [14]). The second one is the cluster-based network.

Table 4-2 shows the comparison. Our proposal stands out because of its higher efficiency in the neighbor selection system (we have added the capacity parameter), lower management cost, high fault tolerance and very high scalability.

4.8 Conclusion

A group-based architecture provides some benefits for the whole network such as the content availability is increased because it could be replicated to other groups, it provides fault tolerance because other groups could carry out tasks from a failed group and it is very scalable because a new group could be added to the system easily. On the other hand, a group-based network can significantly decrease the communication cost between end-hosts by ensuring that a message reaches its destination with small overhead and highly efficient forwarding. Grouping nodes increases the productivity and the performance of the network with low overhead and low extra network traffic.

In this chapter, we have shown a group-based architecture where links between groups could be established by physical proximity plus neighbor node capacity (λ). We have analyzed each role that a node could take on and an analysis is presented to show the efficiency in the neighbor selection process.

Table 4-2: Comparison of routing protocols in group-based topologies.

	Locality-aware overlay network (Z. Xiang et al.)	Cluster Based Topologies	Our group-based proposal
Need of a Rendezvous Point	Yes	No	No
Nodes with higher role	No	Yes	No
Type of topology	Logical, but it could be implemented in physical	Physical and Logical	Physical and logical
Neighbor selection	Proximity in the underlying network (IP)	Physical proximity	Physical proximity + capacity
Which group to join in	Based on rendezvous point decision + boot nodes	Proximity	Based on neighbor discovery (time to reply or closest)
Management cost	Medium because of the rendezvous point	Medium because of cluster head	Low
Fault tolerance	Very low because Rendezvous Point or boot nodes failure	Low because cluster head failure	Very much
Scalability	Very much (depending on the RP)	medium	Very much
Availability	Low (when boot nodes from a group are not available, the group is not available)	Low (when a cluster head is not available, the group is not available)	Very high (when a sensor finds a neighbor it joins the network)

Then, we have presented every message used in this group-based protocol. We have seen every fields needed and when each message is used, explaining some situations.

Its operation, maintenance and fault tolerance have been detailed. Messages designed to work properly have been shown. All simulations show its viability and how it could be designed to improve its performance. Finally, the designed group-based architecture has been compared with other similar architectures. In this comparison we have been able to see that our group-based proposal improves the previous architectures done by other authors.

The architecture proposed could be used for specific cases or environments such when it is wanted to setup a network where groups appear and join the network or by networks that are wanted to be split into smaller zones to support a large number of nodes. But, there are many application areas for this proposal such as rural and agricultural environments or even for military purposes.

Finally, the work presented in this chapter has been published in the following references [20], [19], [13], [12], [18] and [16].

Chapter 5

GROUP-BASED ARCHITECTURE AND PROTOCOL IMPLEMENTATION

5.1 Introduction

Every system is programmed by using a programming model. A programming model is a set of concepts, primitives and combinators to express a computation [92]. A programming language will form part of the programming model, but ‘programming model’ is more general. They can operate on different levels of abstraction. A programming model can for example be made to facilitate programming a single processor, a single system or an entire network. For a programming model to be usable, it should include a programming language and a mapping of a program to operation.

To improve the programmability (and thus deployment and flexibility) of a WSN, effort should be spent on improving its programming model. Programming models used in other types of systems (such as PC’s) are less suited for WSN’s as though in conventional distributed programming individual machines are by themselves powerful systems, this is not true for wireless sensor nodes. Additionally, much more than in regular computer systems the value of a sensor network comes from collaboration among the nodes [93]. For optimal programmability and performance of WSN’s it is necessary to match the concepts of the programming model with those required for a WSN application.

There already exists work on programming models on wireless sensor networks. These have differing goals, methods and levels of abstraction. Some programming models can work on top of each other. Taxonomy of WSN programming models has been shown in [94]. Where authors divide WSN programming models into three groups.

- Operating system: On the lowest level of abstraction, there are the programming models associated with node operating systems. The popular TinyOS operating system is associated with the nesC programming language, which is based on event-based concurrency.
- Support: The supporting programming models are built on top of an operating system to provide the developer with some extra facilities and

generally do not contain any major additional abstractions over nodes or their sensor data.

- Abstraction: The category Abstraction contains the higher-level programming models which provide abstractions over nodes or their sensor data. The most important distinction here is of that between network-centric and node-centric programming models. Network-centric programming, also known as macro-programming, allows "application designers to write code in a high-level language that captures the operation of the sensor network as a whole".

But nowadays it is changing; several manufacturers of wireless sensor nodes are using C as a programming language [95]. In our case, we are going to use C++ because it is the programming language that uses OMNeT++. But, our implementation presented in this chapter could adapt to C easily. So, our goal that was to implement our group-based architecture. It could be used in a real application too, only doing little changes.

This chapter shows the implementation of our group-based protocol. Firstly we present the tools and the programming language used. We explain the OMNet++ simulator, which are its features, its modeling concepts and how it works. Then, we pay attention to C++ programming language. We describe the benefits of object-oriented programming languages and before we show a briefly information about C++. In section 5.3 are explained every class used in our implementation. We have used UML diagrams to show the classes' relation. This implementation is checked in several simulations and we present them in section 5.4. Finally, our conclusions about this chapter are detailed in the last section of this chapter.

5.2 Tools and Programming Language

In this section we are going to show tools and programing languages used to implement our group-based proposal. Firstly, we will explain OMNeT++, which is the simulation tool used to check our group-based protocol. C++ is the main programming language used to develop our proposal.

5.2.1 OMNeT++

In this subsection we are going to talk about OMNeT++. We will see its most important features and how it is organized. All this information is in its website [96]. Moreover there two manuals [97] and [98], which explain in detail the information needed to develop a protocol in this network simulator.

OMNeT++ is an object-oriented modular discrete event network simulation framework. It has a generic architecture, so it can be (and has been) used in various problem domains:

- Modeling of wired and wireless communication networks.
- Protocol modeling.
- Modeling of queuing networks.

- Modeling of multiprocessors and other distributed hardware systems.
- Validating of hardware architectures.
- Evaluating performance aspects of complex software systems.
- In general, modeling and simulation of any system where the discrete event approach is suitable, and can be conveniently mapped into entities communicating by exchanging messages.

5.2.1.1 *Modeling Concepts*

An OMNeT++ model consists of modules that communicate with message passing. The active modules are termed simple modules; they are written in C++, using the simulation class library. Simple modules can be grouped into compound modules and so forth; the number of hierarchy levels is unlimited. The whole model, called network in OMNeT++, is itself a compound module. Messages can be sent either via connections that span modules or directly to other modules. The concept of simple and compound modules is similar to DEVS atomic and coupled models.

In **Figure 5-1**, boxes represent simple modules (gray background) and compound modules. Arrows connecting small boxes represent connections and gates.

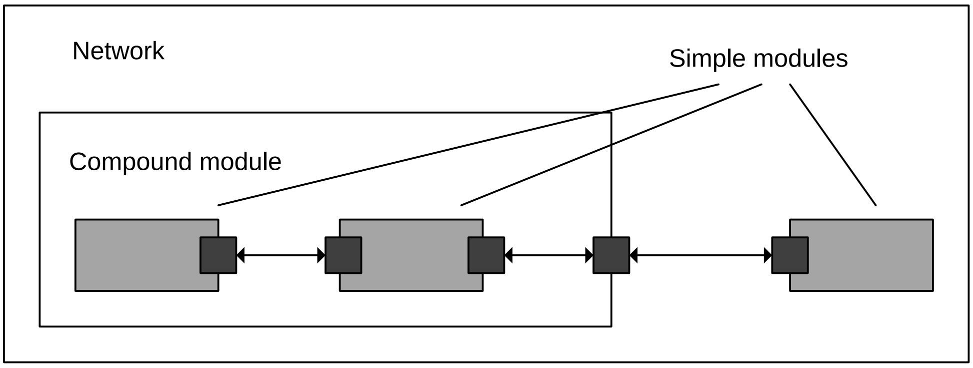


Figure 5-1: Components in OMNeT++.

Modules communicate with messages that may contain arbitrary data, in addition to usual attributes such as a timestamp. Simple modules typically send messages via gates, but it is also possible to send them directly to their destination modules. Gates are the input and output interfaces of modules: messages are sent through output gates and arrive through input gates. An input gate and output gate can be linked by a connection. Connections are created within a single level of module hierarchy; within a compound module, corresponding gates of two submodules, or a gate of one submodule and a gate of the compound module can be connected. Connections spanning hierarchy levels are not permitted, as they would hinder model reuse. Because of the hierarchical structure of the model, messages typically travel through a chain of connections, starting and arriving in simple modules. Compound modules act like "cardboard boxes" in the model, transparently relaying messages between their inner realm and the outside world. Parameters such as propagation delay, data rate and bit error rate, can be assigned to connections. One can also define connection types with specific properties (termed channels) and reuse them in several places. Modules can have parameters. Parameters are used mainly to pass configuration data to simple modules, and to help define model topology. Compound modules may pass parameters or expressions of parameters to their submodules (see an example in **Figure 5-2**).

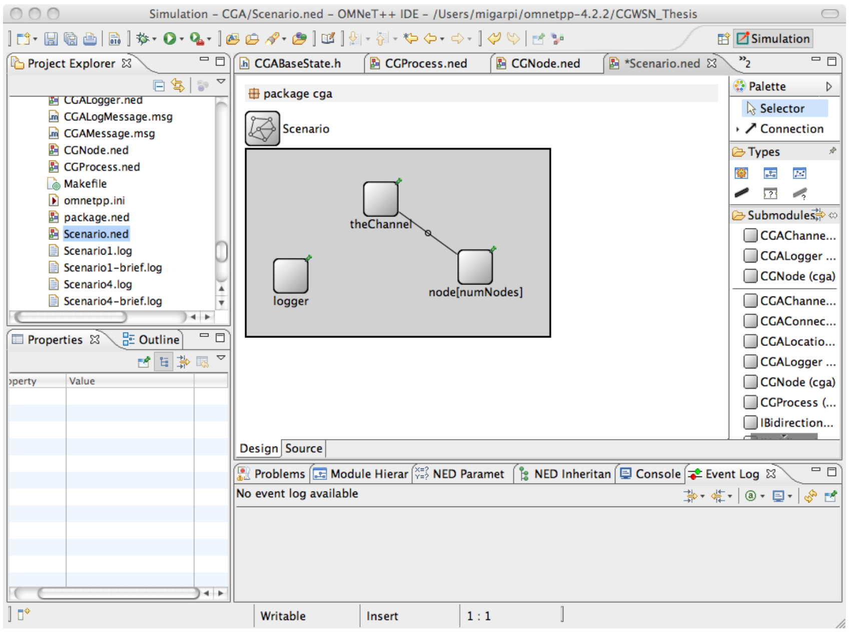


Figure 5-2: Example of modules in OMNeT++.

OMNeT++ provides efficient tools for the user to describe the structure of the actual system. Some of the main features are the following:

- Hierarchically nested modules.
- Modules are instances of module types.
- Modules communicate with messages through channels.
- Flexible module parameters.
- Topology description language.

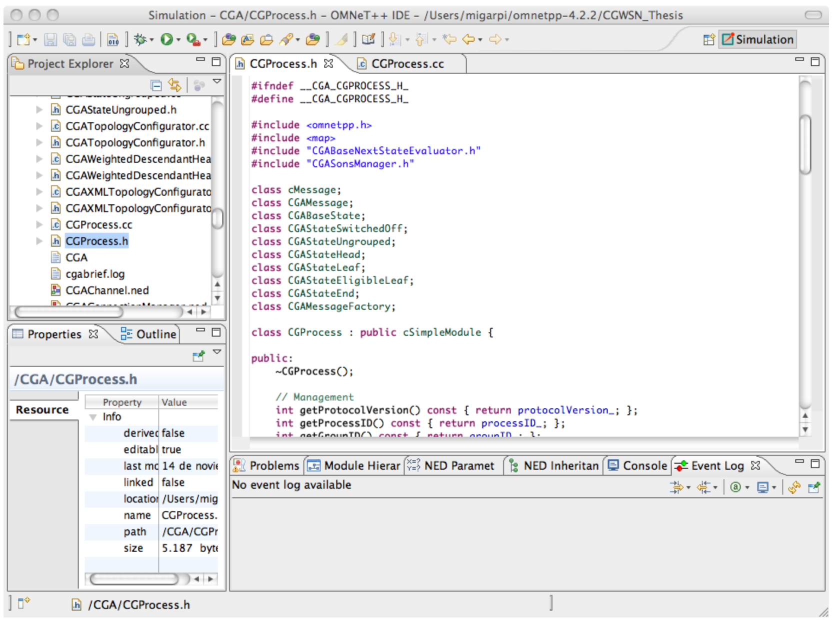
5.2.1.2 *Programming the Algorithms*

The simple modules of a model contain algorithms as C++ functions. The full flexibility and power of the programming language can be used, supported by the OMNeT++ simulation class library. The simulation programmer can choose between event-driven and processstyle description, and freely use object-oriented concepts (inheritance, polymorphism etc) and design patterns to extend the functionality of the simulator.

Simulation objects (messages, modules, queues etc.) are represented by C++ classes. They have been designed to work together efficiently, creating a powerful simulation programming framework. The following classes are part of the simulation class library:

- Module, gate, parameter, channel
- Message, packet
- Container classes (e.g. queue, array)
- Data collection classes
- Statistic and distribution estimation classes (histograms, P2 algorithm for calculating quantiles etc.)
- Transient detection and result accuracy detection classes

The classes are also specially instrumented, allowing one to traverse objects of a running simulation and display information about them such as name, class name, state variables or contents. This feature makes it possible to create a simulation GUI where all internals of the simulation are visible (see **Figure 5-3**).



```
#ifndef __CGA_CGPROCESS_H_
#define __CGA_CGPROCESS_H_

#include <omnetpp.h>
#include <map>
#include "CGABaseNextStateEvaluator.h"
#include "CGASonsManager.h"

class cMessage;
class CGAMessage;
class CGABaseState;
class CGAStateSwitchedOff;
class CGAStateUngrouped;
class CGAStateHead;
class CGAStateLeaf;
class CGAStateEligibleLeaf;
class CGAStateEnd;
class CGAMessageFactory;

class CGProcess : public cSimpleModule {

public:
    ~CGProcess();

    // Management
    int getProtocolVersion() const { return protocolVersion_; };
    int getProcessID() const { return processID_; };
```

Property	Value
Info	
derivec	false
editabl	true
last mc	14 de novie
linked	false
locatior	/Users/mig
name	CGProcess.
path	/CGA/CGPr
size	5.187 byte

```
No event log available
```

Figure 5-3: GUI of OMNeT++.

5.3 Structure of Group-based Architecture Implementation

In this section we explain the main classes that we have implemented in order to simulate the protocol described in the previous chapter. First we must indicate that the language used was C++ and we have helped ourselves using OMNeT++ simulator. All lower layers details have been left to OMNeT++ library.

Our group management protocol is located above the transport layer. For this reason we used a simulator that is very helpful to us when simulating the rest of existing layers. In **Figure 5-5** we can see an overview of relations between the most important classes of our implementation. Object-oriented programming allows us a modular programming, where the use of different interconnected classes could give us the final implementation. This modularity is very helpful to us for improving or introducing new contributions to the project.

CGProcess is the main class, which controls the entire process of creation and maintenance of the groups in our group-based architecture, it is a subclass of. cSimpleModule, a component given by OMNeT++. This class manages its state by means of several subclasses of CGABaseState class. Each of these subclasses implements state-depended functionality. These classes are: CGAStateSwitchedOff, CGAStateUngrouped, CGAStateHead, CGAStateLeaf, CGAStateEligibleLeaf and CGAStateEnd. Their names are related to every state, except CGAStateSwitchedOff that it is a pseudo-state used to simulate inactive nodes.

This class has several associated classes, such as: CGABaseState, CGAStateUngrouped, CGAStateHead, CGAStateLeaf, CGAStateEligibleLeaf, CGAStateEnd, CGAMessageFactory, cMessage, CGA-Message and CGABaseNextStateEvaluator. In **Figure 5-4** we can see the header of CGProcess class. Here we have all the associated classes.

```
#include <omnetpp.h>
#include <map>
#include "CGABaseNextStateEvaluator.h"
#include "CGASonsManager.h"

class cMessage;
class CGAMessage;
class CGABaseState;
class CGAStateSwitchedOff;
class CGAStateUngrouped;
class CGAStateHead;
class CGAStateLeaf;
class CGAStateEligibleLeaf;
class CGAStateEnd;
class CGAMessageFactory;

class CGProcess : public cSimpleModule {
...

};
```

Figure 5-4: CGProcess class header.

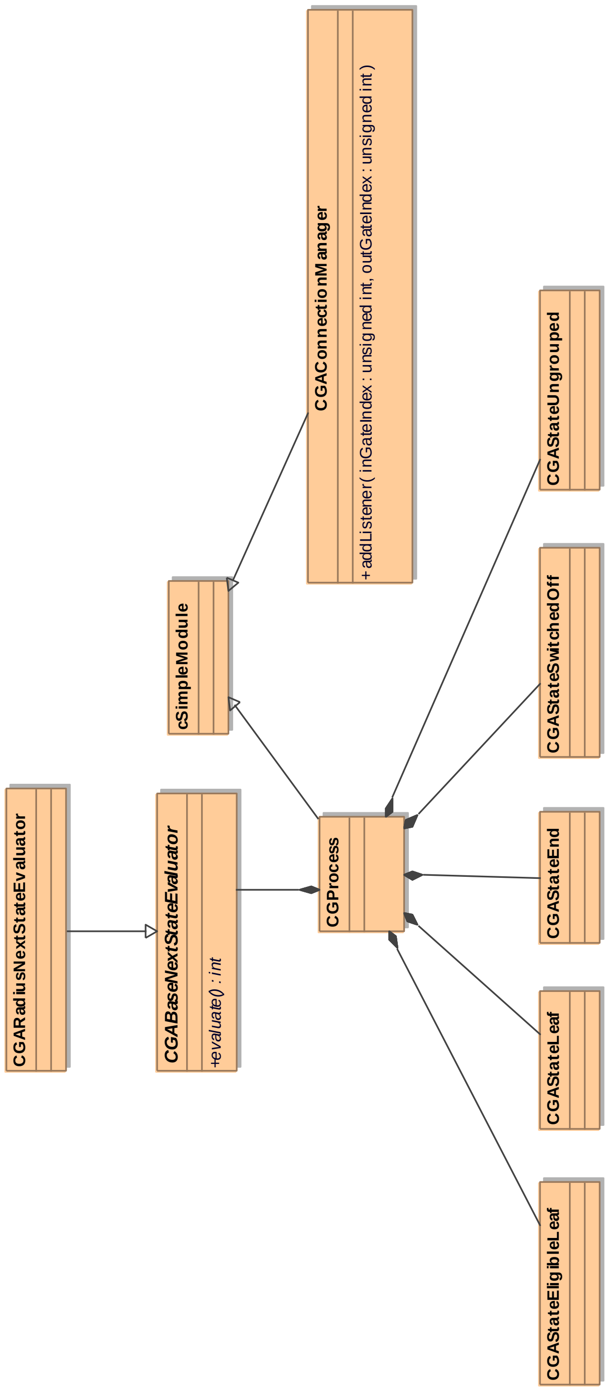


Figure 5-5: General structure view

In the implementation process we have created four abstract classes. CGABaseACKManager, CGABaseState, CGABaseHeadChangeEvaluator and CGANextStatedEvaluator. These abstract classes implement generic methods that all related classes can use or modify. In **Figure 5-5** we can see some examples.

In **Figure 5-6** we can see CGABaseState class, it is an abstract class. This class declares interface, and a few implementations, of the methods that subclasess must implement. . Figure 2-7 shows the class header file. The methods ended with “=0” are abstract and they must be implemented in subclasses. The virtual methods use to offer a default implementation that can be reused by subclasses or they can be overrided if different behavior from default is required. This class implements various generic methods (it is handle message). Some of these methods will be used for generalized classes CGAStateUngrouped, CGAStateHead, CGAStateLeaf, CGAStateEligibleLeaf and CGAStateEnd. But most of them undertake some changes in some methods, one example can be seen in **Figure 5-7**.

```
#include <cobject.h>
#include "CGProcess.h"

class CGAMessage;
class CGABaseState : public cObject {

public:
   CGABaseState(CGProcess *process,const char *displayString = "",int
stateID = 0);
   virtual ~CGABaseState();
   virtual void handle(cMessage *msg) = 0;
   virtual void resetState() = 0;
   virtual void updateIcon();
// Message handling
   virtual void handleHelloMessage(CGAMessage *message);
   virtual void handleACKMessage(CGAMessage *message);
   virtual void handleFatherChosenMessage(CGAMessage *message);
   virtual void handleNewSonInformationMessage(CGAMessage *message);
   virtual void handleAbandonGroupMessage(CGAMessage *message);
   virtual void handleNewHeadMessage(CGAMessage *message);
   virtual void handleStillAliveMessage(CGAMessage *message);
   virtual void handleYetAliveMessage(CGAMessage *message);
   virtual void handleDescendantLeftMessage(CGAMessage *message);
   virtual void handleHeadChangeMessage(CGAMessage *message);
   virtual void handleUnexpectedMessage(CGAMessage *message);
   void setDisplayString(const char *displayString);
   const char *getDisplayString() const;
   int getStateID() const { return stateID_;  };

protected:
   CGProcess *mainProcess_;
   cDisplayString stateDisplayString_;
   int stateID_;
};
```

Figure 5-6: CGABaseState class header.

CGAStateUngrouped class is responsible for managing the HelloACKmessages. Through this class we can manage the sequence numbers, order of arrival etc. The class has an ACK Manager that chooses the best received ACK message, if any. We have implemented two ACK Manger:: a) CGAFirstACKManager, it is a very simple implementation where the node selected as most suitable is the first to reply b) CGAPriorityACKManager, this implementation is the algorithm is presented in Chapter 4 and it takes into account the value of λ.

CGAStateHead manages head change with the help of a head change evaluator modeled through the class CGABaseHeadChangeEvaluator. This class is an abstract class that must be subclassed to implement the offered functionality. Three subclass implementations have been written (see **Figure 5-8**). Two of them where the value to be considered for a change of head is the number of descendants and one that takes into account the weight of each branch of the head node. This latest deployment is better than the first one, because it takes into account whether descendants are very close to the head node or on the contrary these descendants are too far from head node in number of hops.

```
#include "CGABaseState.h"

class CGAStateEligibleLeaf: public CGABaseState {
public:
    CGAStateEligibleLeaf(CGProcess *process,const char *displayString =
"");
    virtual ~CGAStateEligibleLeaf();
    virtual void handle(cMessage *msg);
    virtual void resetState();
    virtual void handleHelloMessage(CGAMessage *message);
    virtual void handleFatherChosenMessage(CGAMessage *message);
    virtual void handleNewHeadMessage(CGAMessage *message);
    virtual void handleHeadChangeMessage(CGAMessage *message);

};
```

Figure 5-7: CGAStateEligibleLeaf class header.

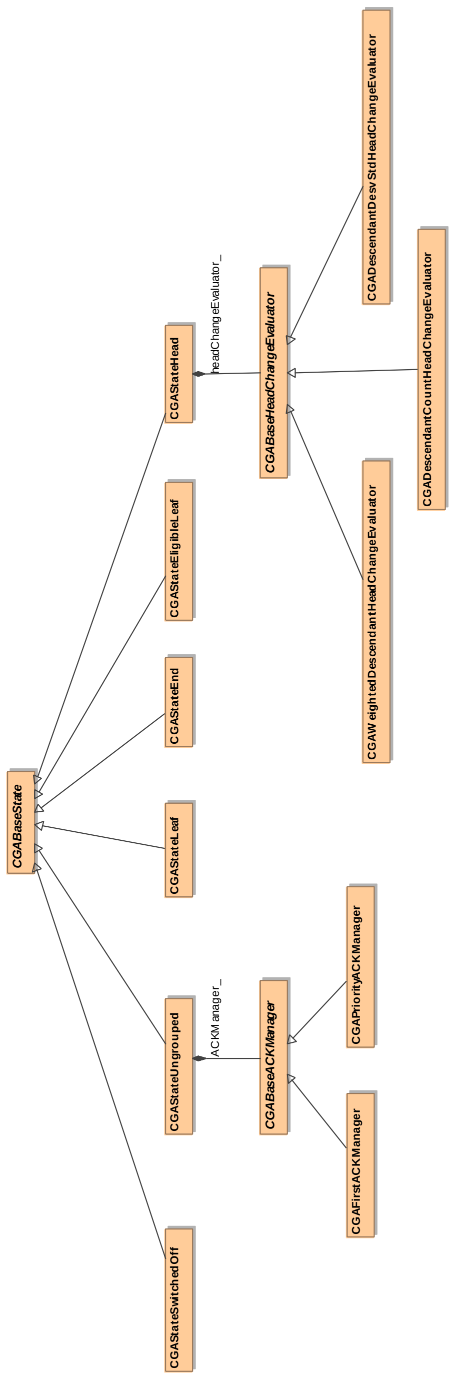


Figure 5-8: State structure view.

CGAStateLeaf, CGAStateEligibleLeaf and CGAStateEnd, the rest of derived classes from CGABaseState that implement their respective state behaviors.

The next CGAMessage class is a subclass of cPacket (see **Figure 5-9**). cPacket is a subclass of cMessage that can be used to represent packets (frames, datagrams, application messages, etc) with OMNeT++. cPacket adds length (measured in bits or bytes), bit error flag, and encapsulation capability to cMessage. CGAMessage models our protocol messages.

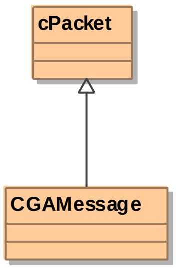


Figure 5-9: Message structure view.

In the implementation of our messages we used the following structure (see **Figure 5-10**). It represents all fields that may have given our messages, which are defined in subsection 4.4.

```
packet CGAMessage {
    int protocolVersion; // For protocol version identification
    int commandID;       // Command identifier for the message
    int srcID;           // ID from source process
    int dstID;           // ID for destination process.
    int sequenceNumber;  // Random sequence number for identify
previous request
    int groupID;        // Group ID to which the process belongs
    int headID;         // Group HEAD Node ID
    int headDistance;   // Number of hops from the process to the Head
    int nodeType;       // What kind of node the process is: Head,
Leaf, Eligible Leaf, etc.
    int extraData;      // Additional data
    int extraData2;     // Additional data
  }
```

Figure 5-10: Content of CGAMessage.msg.

Finally, we want to show our topology generator, which has been implemented in this thesis (see **Figure 5-11**). In order to implement this topology generator we used an abstract class called CGATopologyConfigurator. From this class we have derived 4 subclasses. CGALinearTopologyConfigurator creates a linear topology. CGARingTopologyConfigurator does the same but in this case creates a ring topology. CGAMeshTopology, which using the number of rows and columns creates a mesh topology. Finally CGAXMLTopologyConfigurator implements a configuration from an XML file (it could be a irregular topology). In that XML file is included the position of each node.

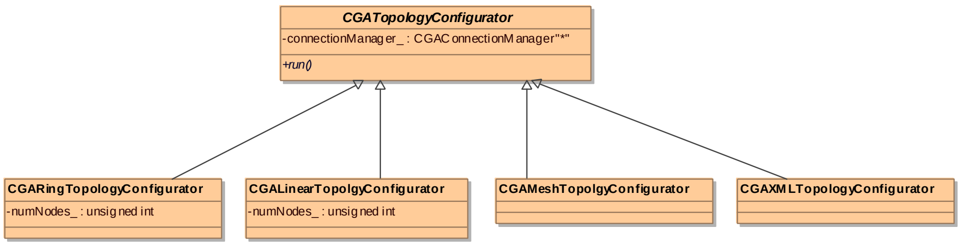

CGATopologyConfigurator
-connectionManager_ : CGAConnectionManager"*"
+run()
CGARingTopologyConfigurator
-numNodes_ : unsigned int
CGALinearTopolgyConfigurator
-numNodes_ : unsigned int
CGAMeshTopolgyConfigurator
CGAXMLTopologyConfigurator

Figure 5-11: Topology configurator structure view.

The topology configurator sets the relation of neighbours of every node at the channel's connection manager. This configurator is executed when the channel is created. The header of this class can be seen in **Figure 5-12**.

```
#include <omnetpp.h>
#include <list>
#include <vector>
#include <string>
#include "CGATopologyConfigurator.h"

class CGAConnectionManager : public cSimpleModule {
protected:
    virtual void initialize();
    virtual void handleMessage(cMessage *msg);

public:
    void addListener(unsigned inGateIndex,unsigned outGateIndex);
};
```

Figure 5-12: CGAConnectionManager class header.

5.4 Implementation's Test

In this section we are going to test our group-based protocol implementation. Firstly, we present our test bench. We will explain how this simulator works with some screenshots. Then, we will a log file in order to check that some behaviors are done correctly. Finally, we will show some results about how many packets are sent, control traffic needed, how many node types are when a topology converges.

At this point, we present the simulator where we have done several testing of our implementation. The simulator appearance can be seen in **Figure 5-13** and **Figure 5-14**. In **Figure 5-13** we can see a scenario in the middle of a simulation. In this case we are simulating a 6x4 mesh network, where there are 24 nodes. The maximum radius of the group is 3 hops between the head node and end nodes. The head nodes are yellow nodes, green nodes are leaf nodes, end nodes are blue and finally the red nodes are eligible leaf nodes. When a node is switched off we represent it in black and when it is ungrouped in white. We can see that all nodes are connected to a module. This module is the channel, this does not mean that all nodes are connected to each other, but they share the same channel. Also this module is connected to another module that is a logger module. This logger module is responsible for collecting all the information. Then we can extract this information from group-based protocol to analyze it.

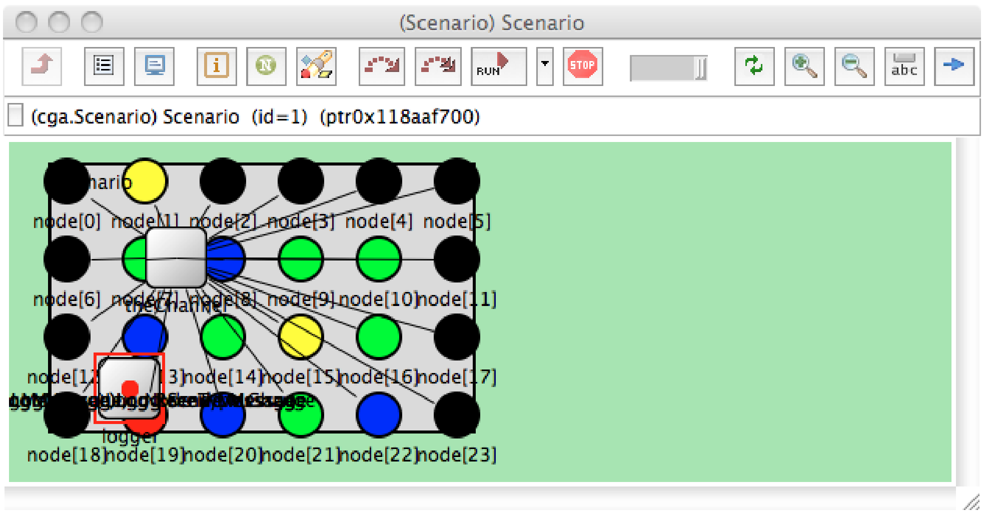
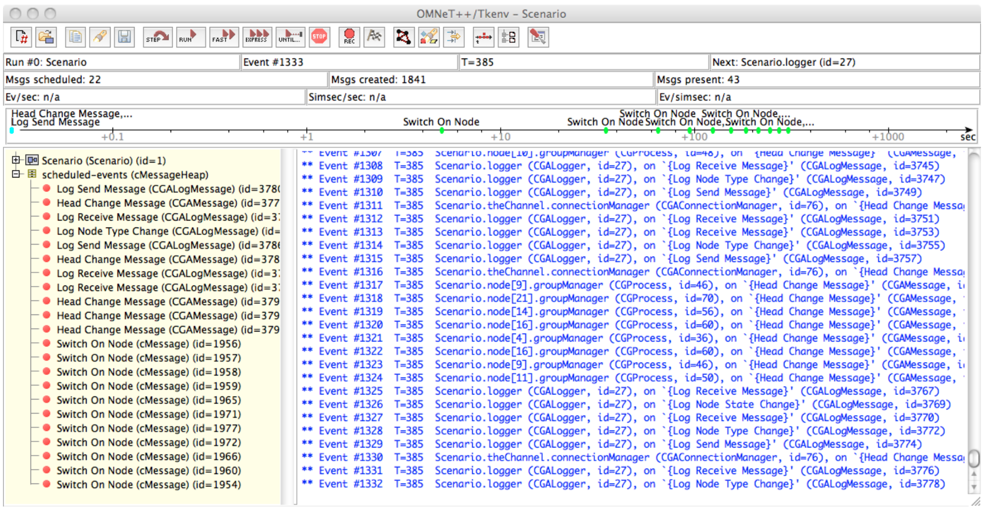


Figure 5-13: Simulator GUI - 1.

The other simulator window is shown in **Figure 5-14**. Here, we highlight three parts. At the top where we can see the ID of the simulation, the running event, the simulation time, the messages created, messages present in the simulation. Furthermore, in the bottom left zone we can see the events that are scheduled. These events are constantly changing. They depend on what is happening on the network. For example, if at any time there is a change of head, all messages related to this change will be planned. Finally in the bottom right, we have an area where we can see all the events that are happening in the network. In this part of the GUI we can see when you send a message, message type, etc. particularly during the simulation process.



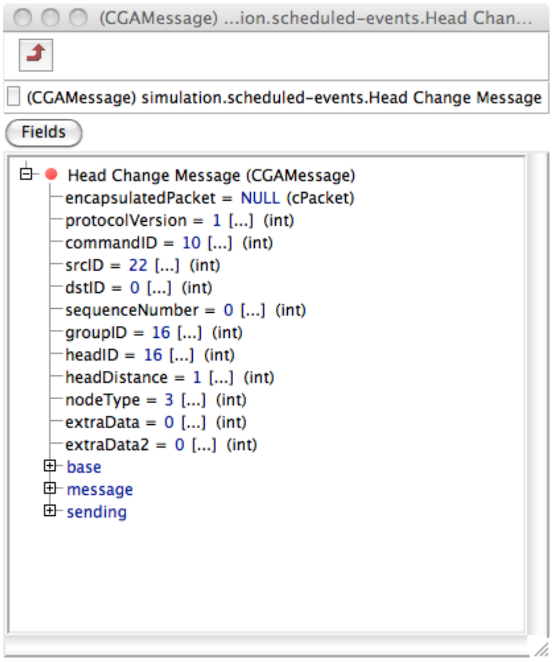
```
Run #0: Scenario            Event #1333            T=385            Next: Scenario.logger (id=27)
Msgs scheduled: 22          Msgs created: 1841          Msgs present: 43
Ev/sec: n/a                 Simsec/sec: n/a             Ev/simsec: n/a
```

```
Scenario (Scenario) (id=1)
scheduled-events (cMessageHeap)
  Log Send Message (CGALogMessage) (id=378
  Head Change Message (CGAMessage) (id=377
  Log Receive Message (CGALogMessage) (id=3
  Log Node Type Change (CGALogMessage) (id=
  Log Send Message (CGALogMessage) (id=378
  Head Change Message (CGAMessage) (id=378
  Log Receive Message (CGALogMessage) (id=3
  Log Receive Message (CGALogMessage) (id=3
  Head Change Message (CGAMessage) (id=379
  Head Change Message (CGAMessage) (id=379
  Head Change Message (CGAMessage) (id=379
  Switch On Node (cMessage) (id=1956)
  Switch On Node (cMessage) (id=1957)
  Switch On Node (cMessage) (id=1958)
  Switch On Node (cMessage) (id=1959)
  Switch On Node (cMessage) (id=1965)
  Switch On Node (cMessage) (id=1971)
  Switch On Node (cMessage) (id=1977)
  Switch On Node (cMessage) (id=1972)
  Switch On Node (cMessage) (id=1966)
  Switch On Node (cMessage) (id=1960)
  Switch On Node (cMessage) (id=1954)
```

```
** Event #1307  T=385  Scenario.node[10].groupManager (CGProcess, id=48), on `{Head Change Message}' (CGAMessage,
** Event #1308  T=385  Scenario.logger (CGALogger, id=27), on `{Log Receive Message}' (CGALogMessage, id=3745)
** Event #1309  T=385  Scenario.logger (CGALogger, id=27), on `{Log Node Type Change}' (CGALogMessage, id=3747)
** Event #1310  T=385  Scenario.logger (CGALogger, id=27), on `{Log Send Message}' (CGALogMessage, id=3749)
** Event #1311  T=385  Scenario.theChannel.connectionManager (CGAConnectionManager, id=76), on `{Head Change Messa
** Event #1312  T=385  Scenario.logger (CGALogger, id=27), on `{Log Receive Message}' (CGALogMessage, id=3751)
** Event #1313  T=385  Scenario.logger (CGALogger, id=27), on `{Log Receive Message}' (CGALogMessage, id=3753)
** Event #1314  T=385  Scenario.logger (CGALogger, id=27), on `{Log Node Type Change}' (CGALogMessage, id=3755)
** Event #1315  T=385  Scenario.logger (CGALogger, id=27), on `{Log Send Message}' (CGALogMessage, id=3757)
** Event #1316  T=385  Scenario.theChannel.connectionManager (CGAConnectionManager, id=76), on `{Head Change Messa
** Event #1317  T=385  Scenario.node[9].groupManager (CGProcess, id=46), on `{Head Change Message}' (CGAMessage, i
** Event #1318  T=385  Scenario.node[21].groupManager (CGProcess, id=70), on `{Head Change Message}' (CGAMessage,
** Event #1319  T=385  Scenario.node[14].groupManager (CGProcess, id=56), on `{Head Change Message}' (CGAMessage,
** Event #1320  T=385  Scenario.node[16].groupManager (CGProcess, id=60), on `{Head Change Message}' (CGAMessage,
** Event #1321  T=385  Scenario.node[4].groupManager (CGProcess, id=36), on `{Head Change Message}' (CGAMessage, i
** Event #1322  T=385  Scenario.node[16].groupManager (CGProcess, id=60), on `{Head Change Message}' (CGAMessage,
** Event #1323  T=385  Scenario.node[9].groupManager (CGProcess, id=46), on `{Head Change Message}' (CGAMessage, i
** Event #1324  T=385  Scenario.node[11].groupManager (CGProcess, id=50), on `{Head Change Message}' (CGAMessage,
** Event #1325  T=385  Scenario.logger (CGALogger, id=27), on `{Log Receive Message}' (CGALogMessage, id=3767)
** Event #1326  T=385  Scenario.logger (CGALogger, id=27), on `{Log Node State Change}' (CGALogMessage, id=3769)
** Event #1327  T=385  Scenario.logger (CGALogger, id=27), on `{Log Receive Message}' (CGALogMessage, id=3770)
** Event #1328  T=385  Scenario.logger (CGALogger, id=27), on `{Log Node Type Change}' (CGALogMessage, id=3772)
** Event #1329  T=385  Scenario.logger (CGALogger, id=27), on `{Log Send Message}' (CGALogMessage, id=3774)
** Event #1330  T=385  Scenario.theChannel.connectionManager (CGAConnectionManager, id=76), on `{Head Change Messa
** Event #1331  T=385  Scenario.logger (CGALogger, id=27), on `{Log Receive Message}' (CGALogMessage, id=3776)
** Event #1332  T=385  Scenario.logger (CGALogger, id=27), on `{Log Node Type Change}' (CGALogMessage, id=3778)
```

Figure 5-14: Simulator GUI - 2.

In **Figure 5-14** if you click on one of the items scheduled, we will get more details. In **Figure 5-15** we see the details of an event when a HeadChange message will be sent. In this example we can see the content of message fields.



```
(CGAMessage) simulation.scheduled-events.Head Change Message

Head Change Message (CGAMessage)
  encapsulatedPacket = NULL (cPacket)
  protocolVersion = 1 [...] (int)
  commandID = 10 [...] (int)
  srcID = 22 [...] (int)
  dstID = 0 [...] (int)
  sequenceNumber = 0 [...] (int)
  groupID = 16 [...] (int)
  headID = 16 [...] (int)
  headDistance = 1 [...] (int)
  nodeType = 3 [...] (int)
  extraData = 0 [...] (int)
  extraData2 = 0 [...] (int)
  base
  message
  sending
```

Figure 5-15: Message's parameters in simulator.

5.4.1 Test Bench

In order to test the implementation of our group-based protocol we performed several simulations, but in this chapter we will present some of them. To obtain the results shown in subsection 5.4.3 we have simulated four scenarios. These scenarios are:

- Lineal Scenario: This scenario is made up of a number of nodes with a linear topology. A node is connected with its neighbors following a bus topology. This topology is a simple topology, but it is very useful for detecting faults. In our case we will create a topology with 9 nodes and we have simulated this scenario in two situations. In the first one, sensor nodes will be activated sequentially every 30 seconds and in the second one, their activation follows an exponential distribution with an average of 10 seconds.
- Ring Scenario: In this scenario the topology followed by nodes is a ring. It is very similar to a linear topology, but in this case the initial node has a connection to the end node. In this case the number of nodes increase to 12 and we have also made two simulations. The first one where the nodes are connected sequentially every 30 seconds and another where the connection follows an exponential distribution with an average of 10 seconds.
- Mesh Scenario: The following scenario is a mesh topology, it consists of 24 nodes. To have this topology we have 6 columns and 4 rows of nodes. The topology of this scenario can be seen in **Figure 5-13**. As we have done with the above scenarios, we will conduct a simulation where nodes are connected sequentially every 30 seconds and another one where the connection is made following an exponential distribution with mean of 10 seconds.
- XML Scenario: Finally we have included a random topology with 100 nodes. This topology is the closest to a real environment because of the number of nodes and their random situation. Like the rest of the above topologies, we carry out a simulation where the nodes are connected sequentially every 30 seconds in three time intervals and another one where the connection of the nodes is performed according to an exponential distribution with mean of 10 seconds.

In all scenarios we set several parameters. The most important are sendHelloTime, helloTimeOut, avoidNAK and maxRadius. All these values are defined in the file omnetpp.ini. An example of this configuration file can be seen in **Figure 5-16**.

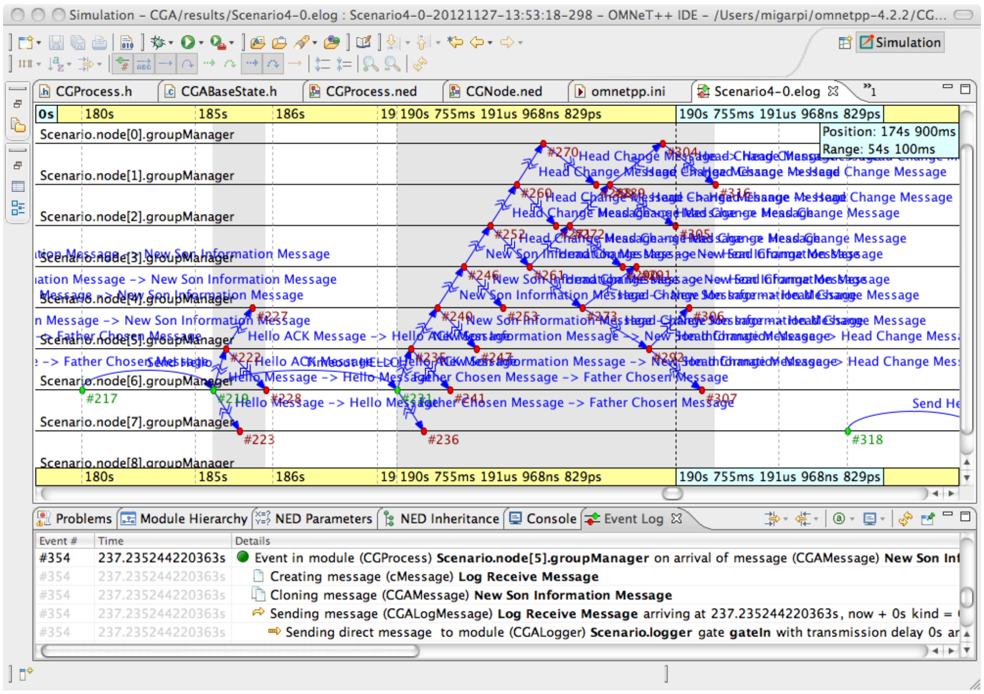
```
[General]
network = Scenario
seed-set = ${runnumber}

[Config Scenario1]
**.numNodes = 9
Scenario.theChannel.connectionManager.topologyGenerator = "Linear"
Scenario.node[*].groupManager.switchOnTime = exponential(10.0s)
Scenario.node[*].groupManager.sendHelloTime = exponential(10.0s)
Scenario.node[*].groupManager.extendedSendHelloTime =
exponential(25.0s)
Scenario.node[*].groupManager.helloTimeout = 5s
Scenario.logger.logFilename = "Scenario1.log"
Scenario.logger.briefFilename = "Scenario1-brief.log"
Scenario.node[*].groupManager.maxRadius = 3
Scenario.node[*].groupManager.headChangeEnabled = true
Scenario.node[*].groupManager.headChangeThreshold = 1.0
```

Figure 5-16: An example of omnetpp.ini.

5.4.2 *Logging Events*

In this subsection we will show several events that happen in our group-based architecture using the logging file of OMNeT++. This file logs all events. The software interprets the file and displays the information using a diagram of events (see **Figure 5-17**).


Simulation - CGA/results/Scenario4-0.elog : Scenario4-0-20121127-13:53:18-298 - OMNeT++ IDE - /Users/migarpi/omnetpp-4.2.2/CG...

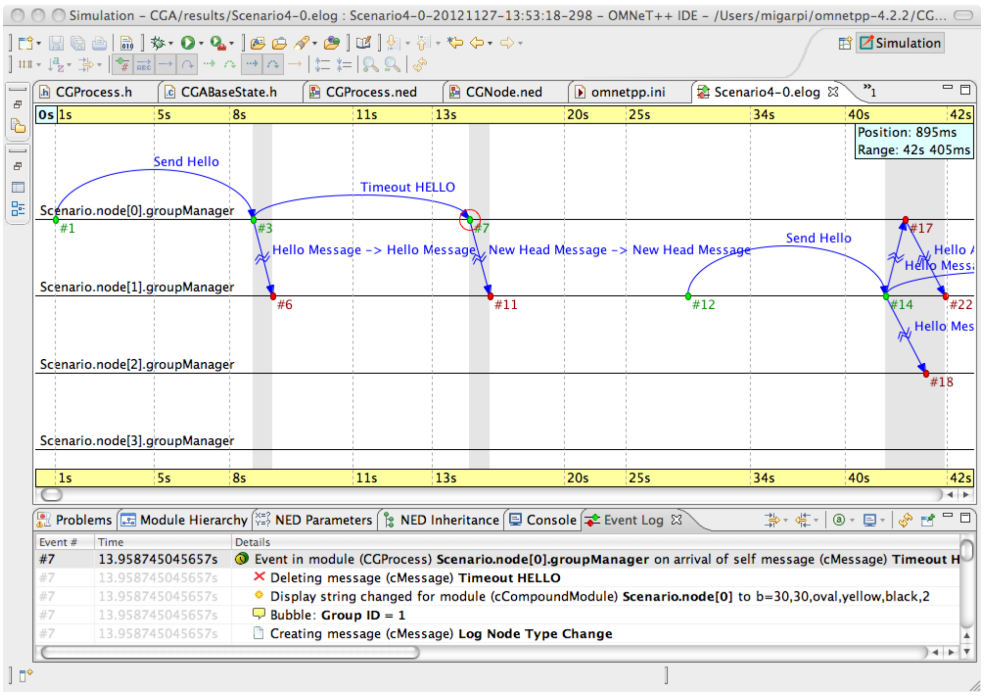
```
Event #  Time               Details
#354     237.235244220363s  Event in module (CGProcess) Scenario.node[5].groupManager on arrival of message (CGAMessage) New Son Inf
#354     237.235244220363s    Creating message (cMessage) Log Receive Message
#354     237.235244220363s    Cloning message (CGAMessage) New Son Information Message
#354     237.235244220363s    Sending message (CGALogMessage) Log Receive Message arriving at 237.235244220363s, now + 0s kind = (
#354     237.235244220363s      Sending direct message  to module (CGALogger) Scenario.logger gate gateIn with transmission delay 0s ar
```
Simulation
CGProcess.h
CGABaseState.h
CGProcess.ned
CGNode.ned
omnetpp.ini
Scenario4-0.elog
0s
180s
185s
186s
190s 755ms 191us 968ns 829ps
Position: 174s 900ms
Range: 54s 100ms
Scenario.node[0].groupManager
Scenario.node[1].groupManager
Scenario.node[2].groupManager
Scenario.node[3].groupManager
Scenario.node[4].groupManager
Scenario.node[5].groupManager
Scenario.node[6].groupManager
Scenario.node[7].groupManager
Scenario.node[8].groupManager
Head Change Message
New Son Information Message
Hello ACK Message -> Hello ACK Message
Hello Message -> Hello Message
Father Chosen Message -> Father Chosen Message
Send Hello
#217
#223
#236
#318
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Event Log

Figure 5-17: Log file view.

5.4.2.1 *Group Creation*

The first event that we will explain using this tool is a group creation. As shown in **Figure 5-18** a node, in this case the node[0], sends a Hello message. This message is sent to the channel, but anyone replies for Hello timeout, so this node creates a group and it becomes head node. When this happens it sends a NewHead message that is not processed by anyone (it is the only node in the network).

After a while a new node appears on the network. This node sends a Hello. This Hello message is replayed by the previous node. This replay will cause that this node sends a FatherChosen indicating that joins the group and also its father is the previous node (with role of Head node).


Simulation - CGA/results/Scenario4-0.elog : Scenario4-0-20121127-13:53:18-298 - OMNeT++ IDE - /Users/migarpi/omnetpp-4.2.2/CG...
Simulation
CGProcess.h
CGABaseState.h
CGProcess.ned
CGNode.ned
omnetpp.ini
Scenario4-0.elog
0s 1s 5s 8s 11s 13s 20s 25s 34s 40s 42s
Position: 895ms
Range: 42s 405ms
Send Hello
Timeout HELLO
Scenario.node[0].groupManager
#1
#3
#7
#17
Hello Message -> Hello Message
New Head Message -> New Head Message
Send Hello
Hello
Hello Mess
Scenario.node[1].groupManager
#6
#11
#12
#14
#22
Hello Mes
Scenario.node[2].groupManager
#18
Scenario.node[3].groupManager
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Event Log

```
Event #  Time               Details
#7       13.958745045657s   Event in module (CGProcess) Scenario.node[0].groupManager on arrival of self message (cMessage) Timeout H
#7       13.958745045657s     Deleting message (cMessage) Timeout HELLO
#7       13.958745045657s     Display string changed for module (cCompoundModule) Scenario.node[0] to b=30,30,oval,yellow,black,2
#7       13.958745045657s     Bubble: Group ID = 1
#7       13.958745045657s     Creating message (cMessage) Log Node Type Change
```

Figure 5-18: Group creation view with log file.

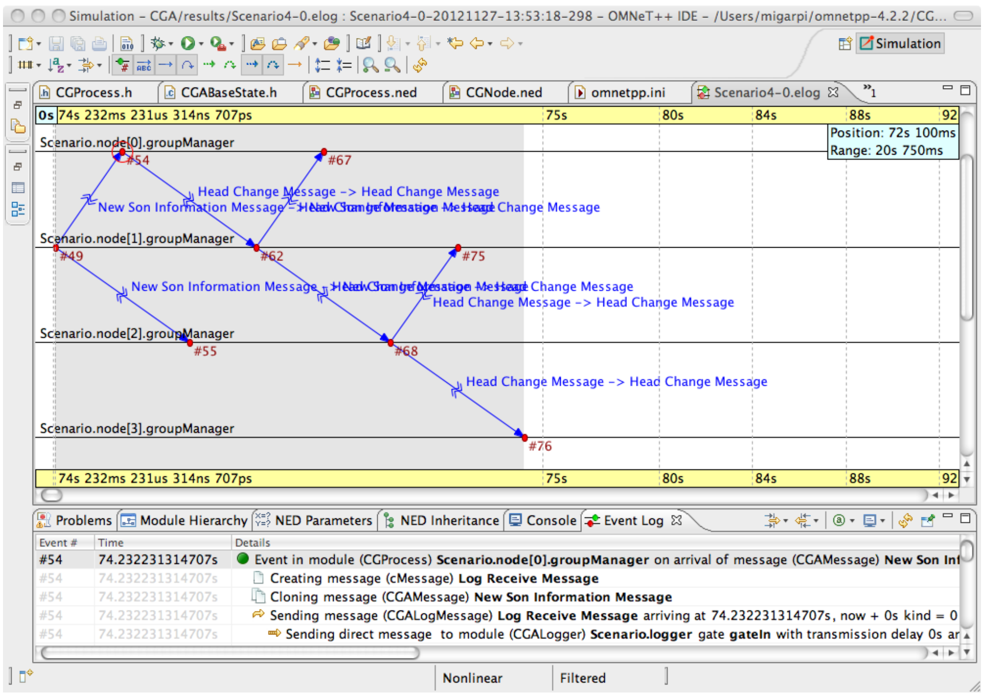
5.4.2.2 *Head Change*

Now, we will explain the process of head change. This process begins when the topology of the group is not centered. When a DescendantLeft or NewSonInformation message arrives to the head node, it is the time that this process can be triggered. In our example (see **Figure 5-19**) everythingstarts when a NewSonInformation message arrives to head node (node[0]).

When node[0] receives a NewSonInformation, it processes the information and verify that its position remains as centered as possible. If not it creates a

HeadChange message with information of new head node. This message is sent to its neighbors and they resend this message with updated information.

In **Figure 5-19**, node[0] sends the message with information of the new head (in this case node[1]). At the same time it sends to node[2] and node[2] to node[3]. Thus all nodes in the group will have the updated information from the head node.


Simulation - CGA/results/Scenario4-0.elog : Scenario4-0-20121127-13:53:18-298 - OMNeT++ IDE - /Users/migarpi/omnetpp-4.2.2/CG...
Simulation
CGProcess.h
CGABaseState.h
CGProcess.ned
CGNode.ned
omnetpp.ini
Scenario4-0.elog
0s 74s 232ms 231us 314ns 707ps
75s
80s
84s
88s
92
Position: 72s 100ms
Range: 20s 750ms
Scenario.node[0].groupManager
#54
#67
Head Change Message -> Head Change Message
New Son Information Message
Head Change Message -> Head Change Message
Scenario.node[1].groupManager
#49
#62
#75
New Son Information Message
Head Change Message -> Head Change Message
Head Change Message -> Head Change Message
Scenario.node[2].groupManager
#55
#68
Head Change Message -> Head Change Message
Scenario.node[3].groupManager
#76
74s 232ms 231us 314ns 707ps
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Event Log
Nonlinear
Filtered

```
Event #  Time                Details
#54      74.232231314707s    Event in module (CGProcess) Scenario.node[0].groupManager on arrival of message (CGAMessage) New Son Inf
#54      74.232231314707s      Creating message (cMessage) Log Receive Message
#54      74.232231314707s      Cloning message (CGAMessage) New Son Information Message
#54      74.232231314707s      Sending message (CGALogMessage) Log Receive Message arriving at 74.232231314707s, now + 0s kind = 0
#54      74.232231314707s        Sending direct message  to module (CGALogger) Scenario.logger gate gateIn with transmission delay 0s ar
```

Figure 5-19: Head change view with log file.

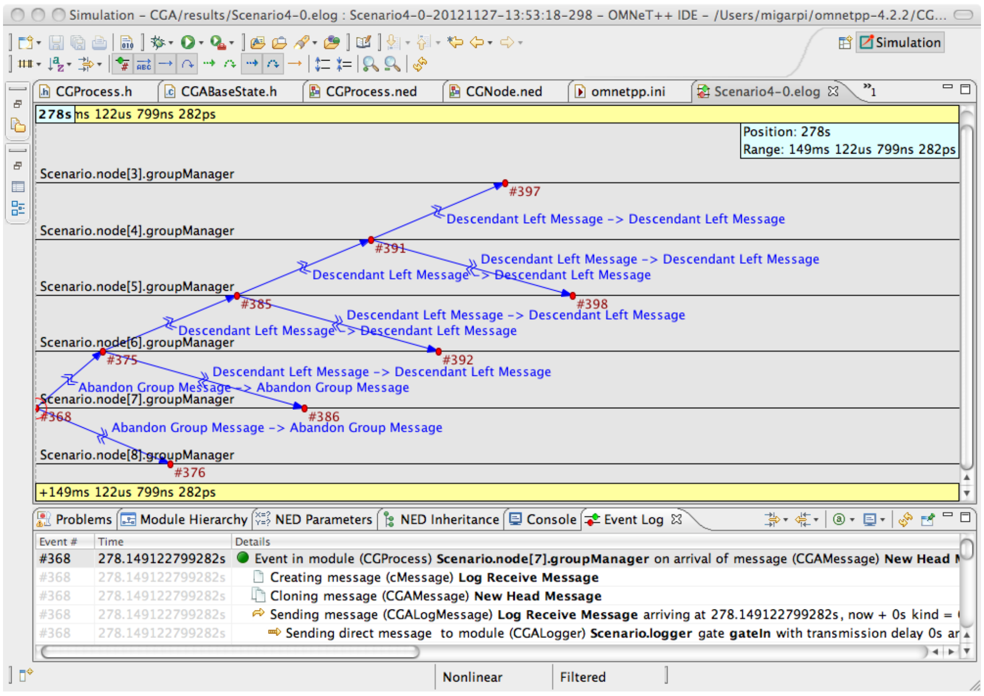
5.4.2.3 <u>*Group Leaving*</u>

Finally, we will explain the process where a node leaves the group to join another that gives a better role. **Figure 5-20** shows an example.

In this case the node[7] has the role eligible leaf when listening a NewHead message from a neighbor node. It decides to leave the group. This action involves sending an AbandonGroup message to its neighbors and they forward the information to their father’s node with a DescendentLeft message. This DescendantLeft will be forward until it arrives at head node. This message contains information to update the network topology and thus having a head node as centered as possible.

Once node[7] leaves the group. It sends a FatherChosen message to the new head to join to this new group. This role will move from eligible leaf to leaf or end, depending on the group design parameters.

When a node leaves a group will be always because it will improve its role in a group.


Simulation - CGA/results/Scenario4-0.elog : Scenario4-0-20121127-13:53:18-298 - OMNeT++ IDE - /Users/migarpi/omnetpp-4.2.2/CG...
Simulation
CGProcess.h
CGABaseState.h
CGProcess.ned
CGNode.ned
omnetpp.ini
Scenario4-0.elog
278s ms 122us 799ns 282ps
Position: 278s
Range: 149ms 122us 799ns 282ps
Scenario.node[3].groupManager
Scenario.node[4].groupManager
Scenario.node[5].groupManager
Scenario.node[6].groupManager
Scenario.node[7].groupManager
Scenario.node[8].groupManager
#397
Descendant Left Message -> Descendant Left Message
#391
Descendant Left Message -> Descendant Left Message
Descendant Left Message -> Descendant Left Message
#385
#398
Descendant Left Message -> Descendant Left Message
Descendant Left Message -> Descendant Left Message
#375
#392
Descendant Left Message -> Descendant Left Message
Abandon Group Message -> Abandon Group Message
#368
#386
Abandon Group Message -> Abandon Group Message
#376
+149ms 122us 799ns 282ps
Problems
Module Hierarchy
NED Parameters
NED Inheritance
Console
Event Log
Nonlinear
Filtered

```
Event #   Time                Details
#368      278.149122799282s   Event in module (CGProcess) Scenario.node[7].groupManager on arrival of message (CGAMessage) New Head M
#368      278.149122799282s     Creating message (cMessage) Log Receive Message
#368      278.149122799282s     Cloning message (CGAMessage) New Head Message
#368      278.149122799282s     Sending message (CGALogMessage) Log Receive Message arriving at 278.149122799282s, now + 0s kind = (
#368      278.149122799282s       Sending direct message  to module (CGALogger) Scenario.logger gate gateIn with transmission delay 0s ar
```

Figure 5-20: Group leaving view with log file.

5.4.3 Results

In this subsection we will show our results from group-based protocol implementation in scenarios described previously. Each topology has an activation process. One is sequential and the other where all nodes are activated following an exponential distribution with a mean of 10 seconds.

5.4.3.1 Lineal Topology

The first analyzed topology is lineal topology. This topology is made of 9 nodes.

Figure 5-21 shows the messages sent to create a group-based architecture in this topology. We can see that the number of messages for both simulations is low. In the Lineal_seq topology, there are 9 Hello messages, the same number as nodes. There are six Hello messages in Lineal_rdm scenario. This number varies between both simulations because 3 ungrouped nodes have listened a NewHead message before sending their respective Hello. So, they have joined the group without sending a Hello.

In the case of Lineal_seq scenario, nodes are activating and sending Hello messages from left to right, so the head node change several times to be as centered as possible. For that reason there are very HeadChange messages in **Figure 5-21**. The number of KeepAlive messages is higher in Lineal_seq scenario because the topology is stable soon and thus KeepAlive message exchange begins earlier.

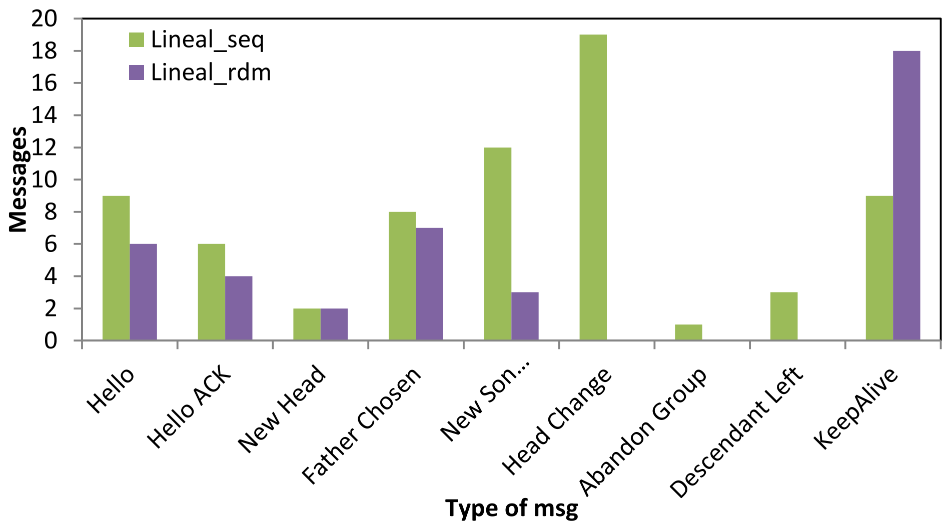


Figure 5-21: Messages needed in lineal scenarios.

In **Figure 5-22** we observe control traffic that injects our group-based protocol to the network. In this topology we can see that the traffic is quite low, and it does not exceed 13 packets.

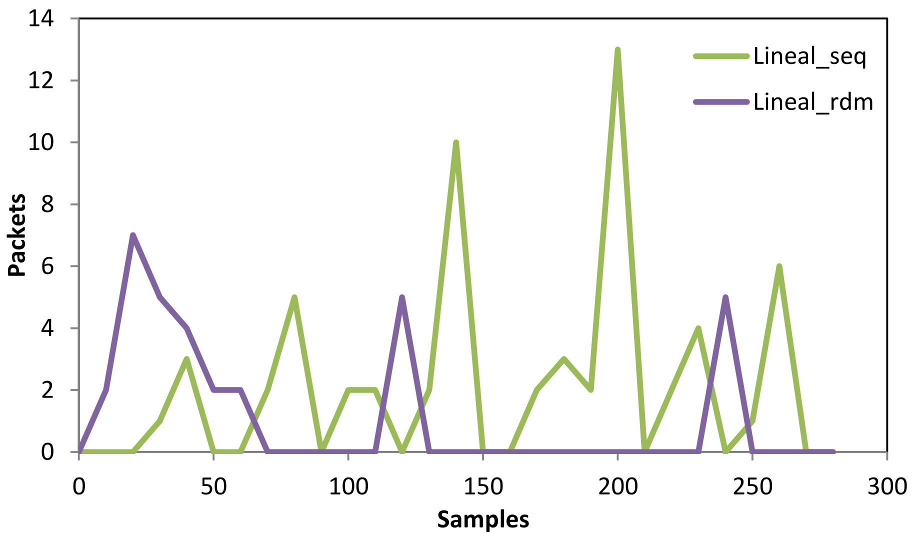


Figure 5-22: Control traffic in lineal scenarios.

We can see that there are differences between both types of node’s activation. For Lineal_seq scenario, control packets are sent during the activation process. This traffic is low and it increases only at specific times. In those moments where there is more traffic because it is executing a head change.

Moreover, the Lineal_rdm simulationl sends much control information at first times because all nodes are already active. During that first moment several Hello, HelloACK and FatherChosen messages are exchanged. For this reason there is more traffic. After the node’s grouping is done it appears KeepAlive exchange to check that nodes are running.

The following figures (**Figure 5-23** and **Figure 5-24**) represent the number of nodes with each roles and their group membership. In **Figure 5-23** we can see that there are two groups (GroupID=1 and GroupID=9) for Lineal_seq simulation. The GroupID=1 has a head node, 4 leaf nodes and two end nodes. In case of GroupID=9, this group only has two nodes, one of them is the head node and other one is leaf node.

In random lineal simulation (see **Figure 5-24**) the number of groups is 2. The group’s IDs are 3 and 7. GroupID=3 has one head node and four leaf nodes. GroupID=7 has one head node and three leaf nodes. In this case, the random activation component and sending Hello messages cause that there are not end nodes.

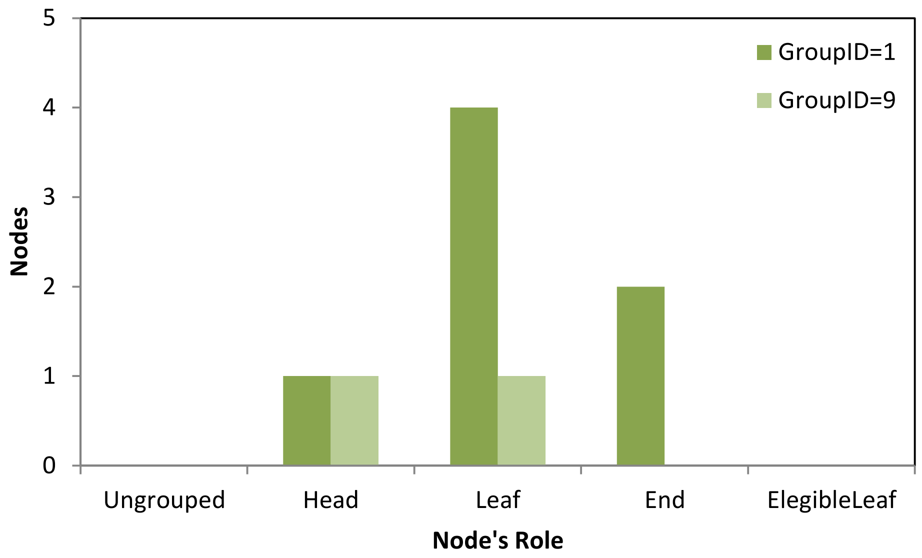


Figure 5-23: Number of nodes in sequential lineal scenario.

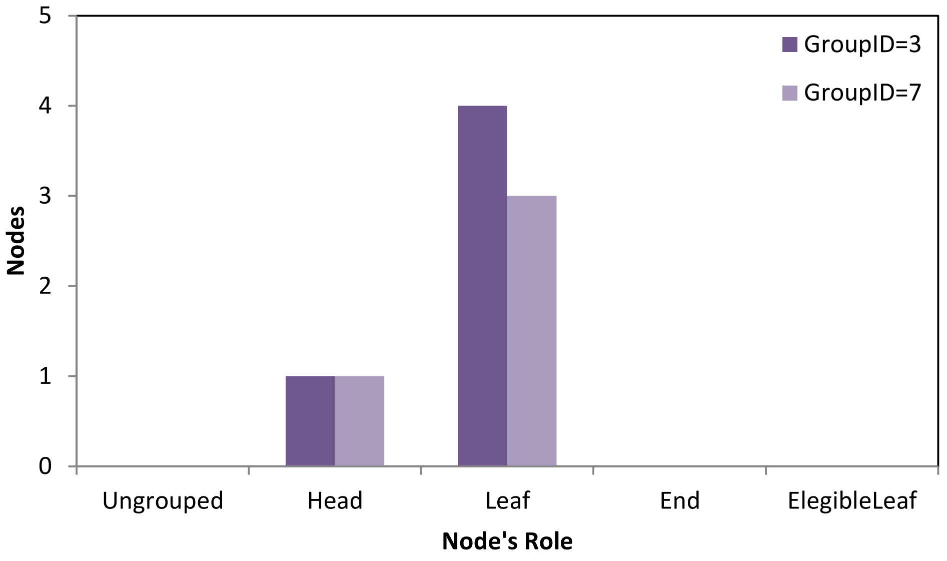


Figure 5-24: Number of nodes in random lineal scenario.

5.4.3.2 *Ring Topology*

Now, we will analyze the results of the ring topology with 12 sensor nodes. Like the previous topology, we have a scenario where the nodes are sequentially activated and another where they are connected following an exponential distribution with a mean of 10 seconds.

Group-based protocol messages are shown in **Figure 5-25**. Here, we can see that the sequential topology starts sending more messages except for KeepAlive messages. In this case, the Ring_rdm simulation starts sending 8 KeepAlive messages more. In the Ring_seq simulation, nodes send 12 Hello messages, the same number as nodes. In case of Ring_rdm, 9 Hello messages are sent, so 3 ungrouped nodes have listened a NewHead message before sending their respective Hello.

Sequential topology causes more head changes than random topology, the same as in **Figure 5-21**. The main reason is that our topology is growing more by a branch than other when our nodes follow a sequential activation, so the head node must change to stand as centered as possible.

Also, we can see that in this Ring_seq scenario there is also a group leaving, which implies that DescendentLeft messages must be sent.

There are more KeepAlive messages in the Ring_rdm scenario because the grouping process is performed in the first moments and then when groups are already created, they begin the process of group's maintaining.

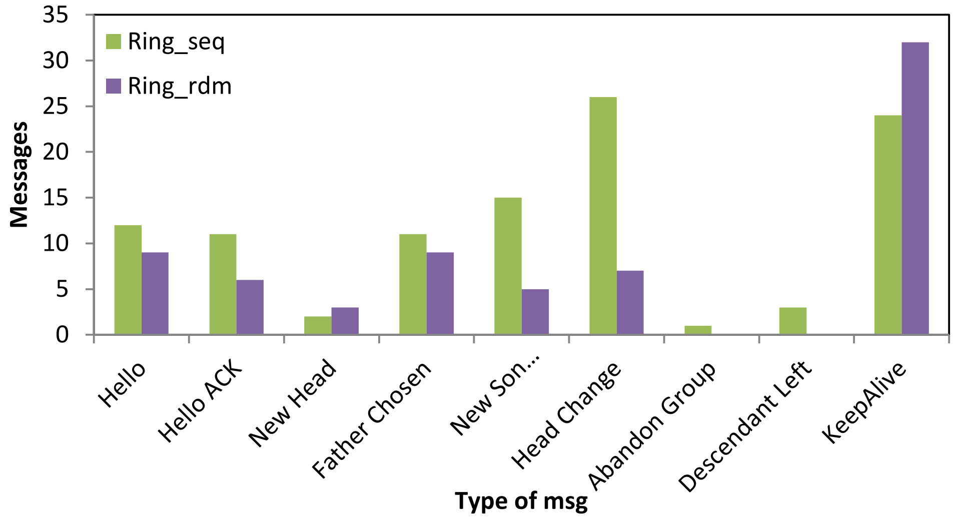


Figure 5-25: Messages needed in ring scenarios.

The control traffic generated by our group-based protocol is represented in **Figure 5-26**. We can see that the traffic generated by the Ring_seq scenario is distributed around all simulation. We have some peaks of 13 packets, at that time there are head changes. Furthermore, in the Ring_rdm scenario there is larger traffic amount at the beginning of the simulation. This is due to the exchange of Hello, FatherChosen and NewSonInformation messages. Then, there are some peaks showing KeepAlive activity.

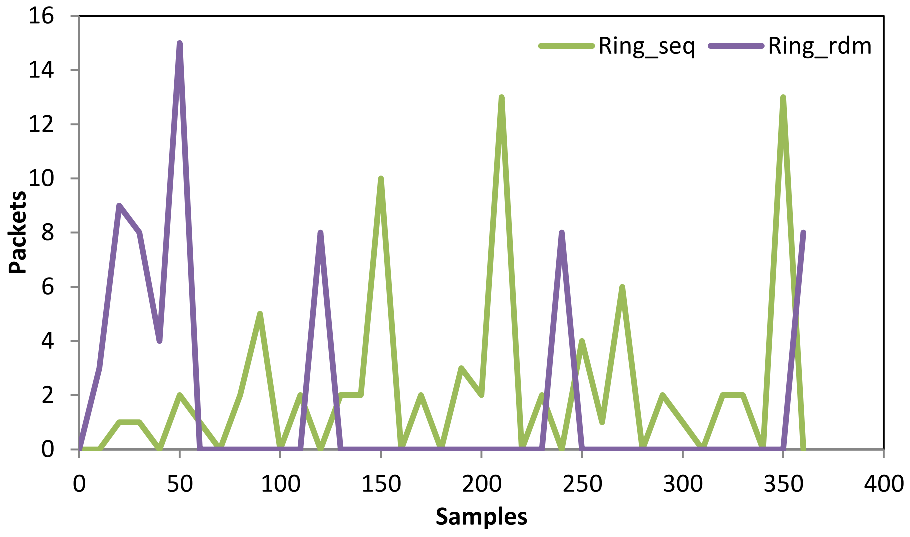


Figure 5-26: Control traffic in ring scenarios.

The following figures (**Figure 5-27** and **Figure 5-28**) are the number of nodes of each type and their group membership. In **Figure 5-27** we can see that there are two groups (GroupID=1 and GroupID=9) for the sequential simulation. GroupID=1 has a head node, 4 leaf nodes and two end nodes. In the case of GroupID=9, this group a head node and 4 leaf nodes.

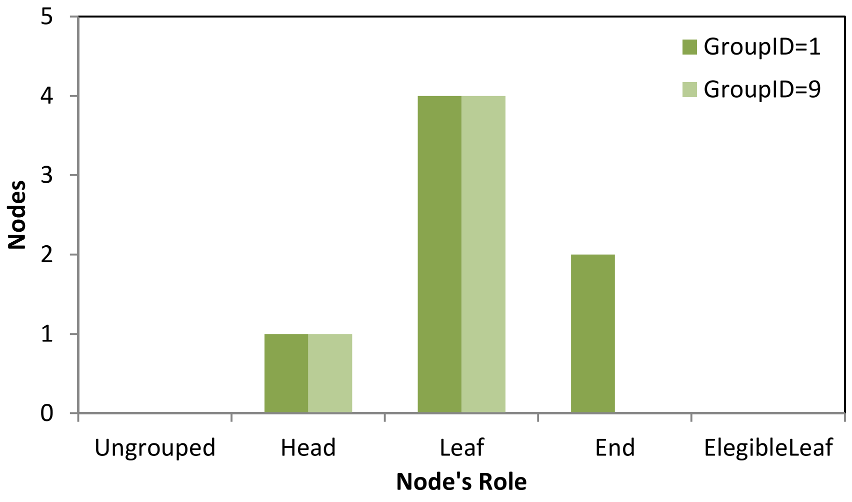


Figure 5-27: Number of nodes in sequential ring scenario.

In the case of random activation (see **Figure 5-28**) the number of groups is 3. The GroupIDs are 1, 3 and 9.The Group 1 consists of a head node and a leaf node. Group 3 has one head node and four leaf nodes. Group 9 has one head node and 4 leaf nodes. In this case, the random activation component and sending Hello messages cause that there are not end nodes.

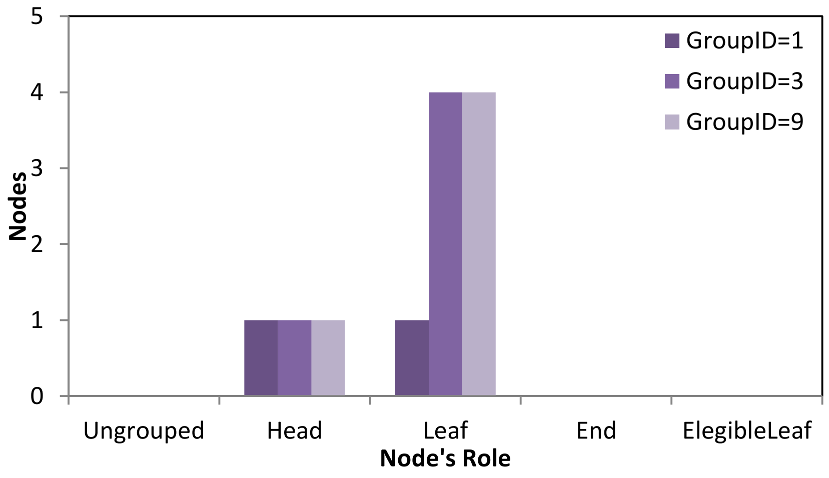


Figure 5-28: Number of nodes in random ring scenario.

5.4.3.3 *Mesh Topology*

In this subsection we will analyze a mesh topology with 24 nodes. In this scenario will have a topology where nodes are activated sequentially, in this case from two non-neighboring sensor nodes to external nodes. In other simulation, node's activation follows an exponential distribution with mean 10 seconds.

Messages sent in these two scenarios are shown in **Figure 5-29**. Here, we can see that the sequential topology nodes start sending more messages except for KeepAlive and NewHead messages. In this case, the Mesh_rdm simulation starts sending 12 KeepAlive messages more. In the Mesh_seq simulation, nodes send 24 Hello messages, the same number as nodes. In case of Mesh_rdm, 12 Hello messages are sent, so 12 ungrouped nodes have listened a NewHead message before sending their respective Hello.

In Mesh_seq topology there are many HeadChange messages. The main reason is that our topology is growing more by a branch than other when our nodes follow a sequential activation, so the head node must change to stand as centered as possible.

In this mesh topology there are several group leavings in both simulations, but the Mesh_seq has more group changes.

If you look at the number of NewHead messages, we can see the number of groups that we have in each scenario. We have 2 groups in the Mesh_seq simulation and 5 groups in the Mesh_rdm simulation.

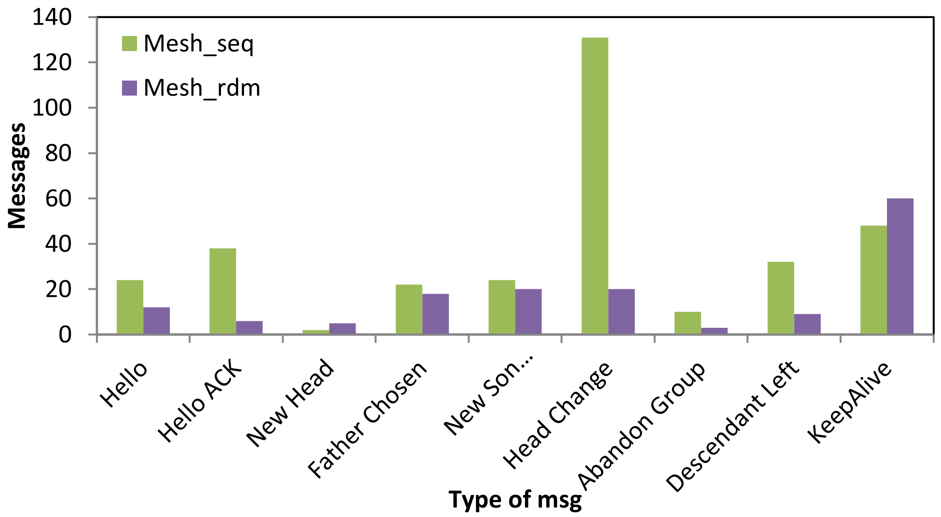


Figure 5-29: Control messages in mesh scenarios.

The control traffic generated by our group-based protocol is represented in **Figure 5-30**. We can see that the traffic generated by the Mesh_seq scenario is distributed around all simulation. We have some peaks of 21 packets, at that time there are head changes. Furthermore, in the Mesh_rdm scenario there is larger traffic amount at the beginning of the simulation. This is due to the exchange of Hello, FatherChosen and NewSonInformation messages. Then, there are some peaks showing KeepAlive activity.

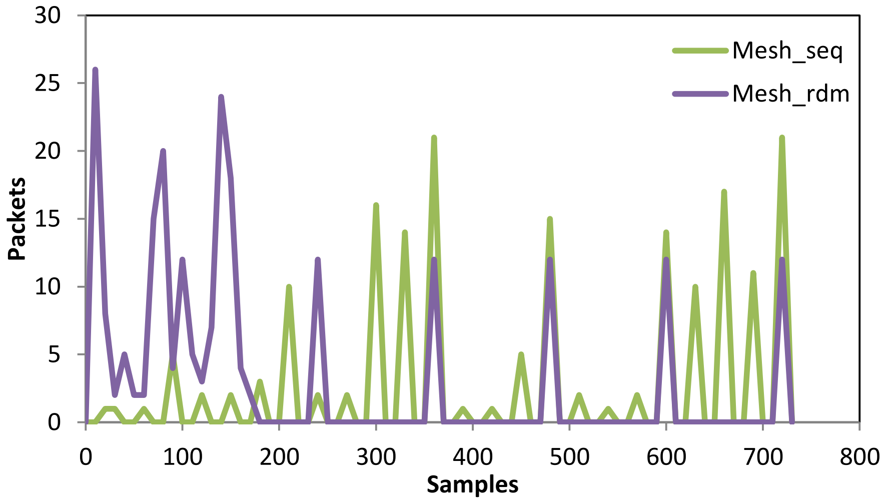


Figure 5-30: Control traffic in mesh scenarios.

The following figures (**Figure 5-31** and **Figure 5-32**) are the number of nodes of each type and their group membership. In **Figure 5-31** we can see that there are two groups (GroupID=8 and GroupID=17) in the simulation with sequential activation. The GroupID=8 has a head node, 8 leaf nodes and 4 end nodes. For the GroupID=17, this group has a head node, 5 leaf nodes, 3 end nodes and 2 eligible leaf nodes.

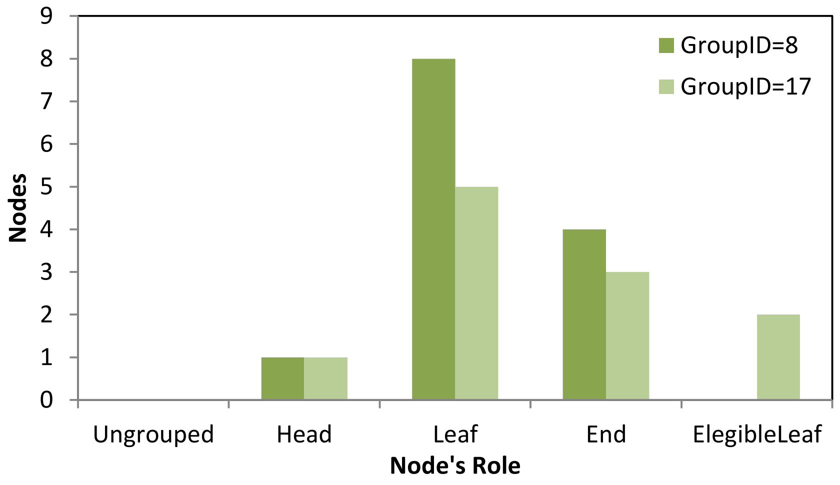


Figure 5-31: Number of nodes in sequential mesh scenarios.

In the case of random activation (see **Figure 5-32**) the number of groups is 5. Group identifiers are 3, 7, 11, 15 and 23. The group with ID=3 is made by a head node, 3 leaf nodes and an end node. Groups 7, 11 and 15 have one head node and 4 leaf nodes each one. The group 23 has one head node, two leaf nodes and one end node. In this case, the random activation component and sending Hello messages cause that there are not end nodes.

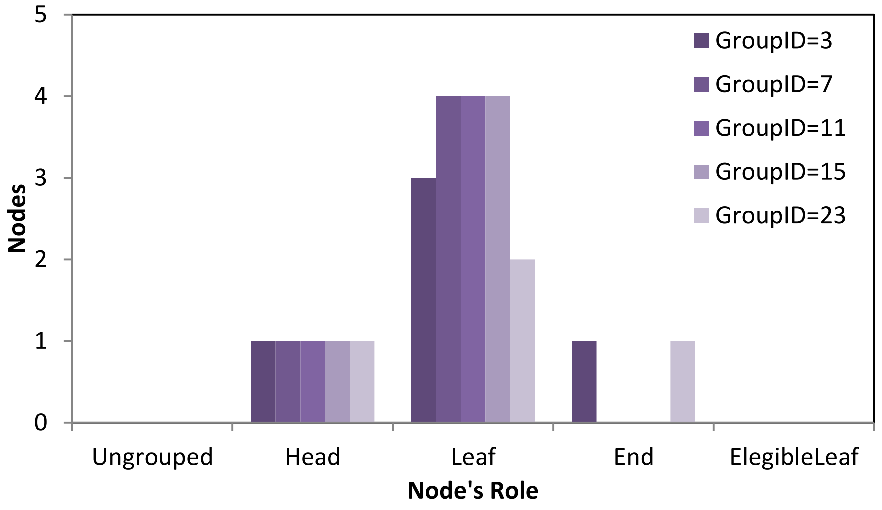


Figure 5-32: Number of nodes in random mesh scenarios

5.4.3.4 *XML Topology*

Finally, we will analyze a random XML topology with 100 nodes. A simulation will have a topology where nodes are being activated sequentially. In another simulation node's activation follows an exponential distribution with a mean of 10 seconds.

Messages sent in these two scenarios are shown in **Figure 5-33**. Here, we can see that the sequential topology nodes start sending more messages except for KeepAlive and NewHead messages. In this case, the XML_rdm simulation starts sending 150 KeepAlive messages more. In the XML_seq simulation, nodes send 96 Hello messages, 4 messages unless the number of nodes. In case of Mesh_rdm, 72 Hello messages are sent, so 28 ungrouped nodes have listened a NewHead message before sending their respective Hello.

In XML_seq topology there are many HeadChange messages. The main reason is that our topology is growing more by a branch than other when our nodes follow a sequential activation, so the head node must change to stand as centered as possible. In this XML topology there are several group leavings in both simulations, but the XML_seq has more group changes.

If you look at the number of NewHead messages, we can see the number of groups that we have in each scenario. We have 12 groups in the XML_seq simulation and 17 groups in the XML_rdm simulation.

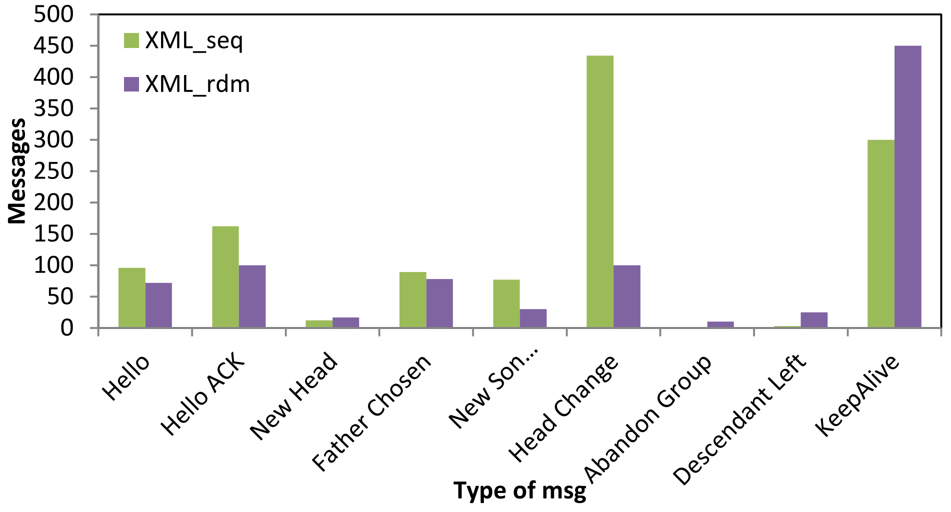


Figure 5-33: Control messages in XML scenarios.

The control traffic generated by our group-based protocol is represented in **Figure 5-30**. We can see that the traffic generated by the XML_seq scenario is distributed around all simulation. We have some peaks of 37 packets, at that time there are head changes. Furthermore, in the XML_rdm scenario there is larger traffic amount at the beginning of the simulation (maximum of 74 packets). This is due to the exchange of Hello, FatherChosen and NewSonInformation messages. Then, there are some peaks showing KeepAlive activity.

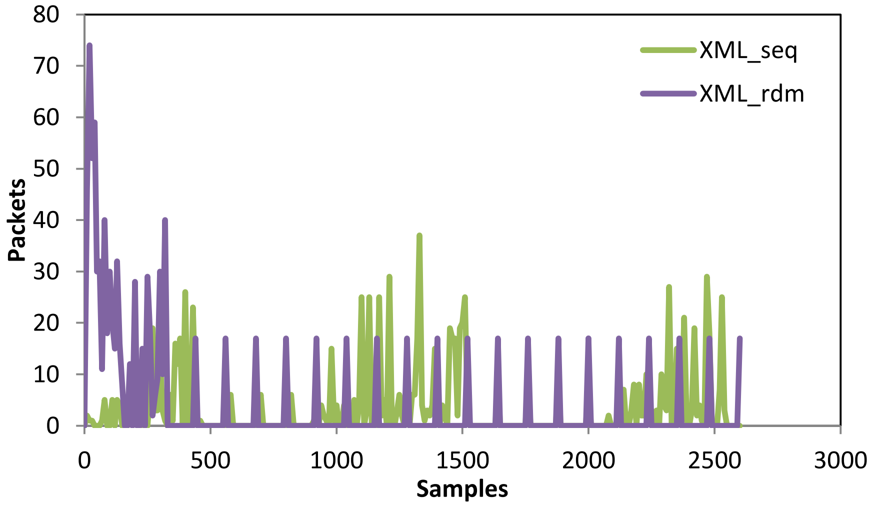


Figure 5-34: Control traffic in XML scenarios.

In the case of XML_seq scenario, we see that there are three time intervals where the control traffic is greater. During these periods of time there are new node's activations and this provides more exchanges of messages.

In **Table 5-1:** Number of nodes in each group (sequencial XML scenario).**Table 5-1** show the groups created in the scenario XML_seq. As we can see there are 12 groups and each of them has an amount of nodes with their functionality. For example the groupID=18 has one head node, 10 leaf nodes, 4 end nodes and ultimately one eligible leaf node. As required by the topology every node has a head group, who manages the group's formation.

Table 5-1: Number of nodes in each group (sequencial XML scenario).

	HEAD	LEAF	END	ELIGIBLE LEAF
Group ID = 1	1	1	0	0
Group ID = 3	1	4	0	0
Group ID = 16	1	5	0	0
Group ID = 18	1	10	4	1
Group ID = 21	1	6	1	0
Group ID = 24	1	5	1	0
Group ID = 51	1	7	2	0
Group ID = 55	1	7	2	0
Group ID = 57	1	7	3	2
Group ID = 86	1	9	2	0
Group ID = 91	1	5	0	0
Group ID = 95	1	4	0	0

Finally we have extracted the same data of XML_rdm simulation. The data are represented in **Table 5-2**. In this table there are 17 groups and each group has a number of nodes with their role. In this case we can see that there are more groups than in the scenario XML_seq, but these groups are more homogeneous.

5.5 Conclusion

In this chapter we have presented the implementation of our group-based protocol to create the architecture proposed for WSNs.

In order to carry out this task first we have presented the tools needed and the programming languages used. To implement this protocol we have used C++ and this was simulated used the OMNeT++ simulator. Through this simulator we have tested our group-based protocol over several layers used sensor networks.

The next task has been to present the implementation of our group-based protocol using UML diagrams. Through these diagrams we have explained the main classes of this implementation. Moreover, with these diagrams we have explained the relationship between the classes and how to implement each of them.

Table 5-2: Number of nodes in each group (random XML scenario).

	HEAD	LEAF	END	ELIGIBLE LEAF
Group ID = 2	1	3	0	0
Group ID = 9	1	2	0	0
Group ID = 14	1	5	1	0
Group ID = 18	1	5	2	1
Group ID = 21	1	4	0	0
Group ID = 26	1	3	0	0
Group ID = 34	1	6	0	0
Group ID = 39	1	2	1	0
Group ID = 48	1	3	0	0
Group ID = 53	1	6	0	0
Group ID = 55	1	3	2	0
Group ID = 66	1	3	1	1
Group ID = 69	1	6	0	0
Group ID = 72	1	5	2	1
Group ID = 86	1	5	2	1
Group ID = 93	1	3	1	0
Group ID = 99	1	3	0	0

Finally, we have presented the results of four network topologies, which implement our group-based protocol. Through these results we have tested that our protocol works well on such networks, it introduces little traffic control and as we have seen in our previous simulation in Chapter 3 that they gives us a high efficiency. It should be noted that our proposal works in a good way when in our topologies we have a node's activation process, which follows an exponential distribution. This feature provides an advantage to this architecture, as we say does not depend on the physical topology of the nodes or their activation process.

The work presented in this chapter has been published in the following references [20] and [19].

Chapter 6

***C**OLLABORATIVE **G**ROUP-BASED **WSN**S*

6.1 Introduction

There are many applications where the WSN could be used. Some examples are: a monitoring system for fire detection [3], for habitat monitoring [4], a fish farm monitoring and control [5], etc.

However, when we want to cover large extensions and areas with difficult access, there are certain technical limitations that require an exhaustive network design, and study, to choose the appropriate communication algorithm and a good network topology. One of the main concerns of the researchers and developers is the whole network power consumption because devices are powered with batteries, and low maintenance is desired.

In order to reduce the power consumption, the hardware design is very important. Some of the techniques employed are related to knowing when the device must be in one mode (active mode, sleep mode or idle mode) or in another [6], thereby achieving to reduce unnecessary power consumption. But it also ensures minimum levels of quality of information flowing through the network. There are many systems, architectures, and protocols that can be used for WSN [7], but in our previous works, we have demonstrated that wireless sensors group-based topologies and networks improve the performance and the efficiency of the whole network [8, 9]. Group-based topologies allow the sensor to operate more flexibly, efficiently and less energy consuming than regular network topologies [10]. Moreover, in one of our previous works we propose a group-based protocol for large wireless ad hoc and sensor networks [11].

This chapter details the improvements achieved by the use of collaborative group-based networks in wireless sensor networks and shows an important mathematical analysis to verify it. WSNs can be used in many real applications (environmental monitoring, habitat monitoring, health, etc.), and as we have seen in previous chapters energy consumption is one of the most critical feature for them. So, achieving a significant energy saving is one of the objectives of this thesis. This is the main improvement achieved from collaborative group-based wireless sensor networks.

Our proposal is to reduce the global energy consumption of WSNs using collaborative based-group networks and it is explained in the third section of this chapter by an analytical model and simulations. Second section compares the regular architectures versus collaborative group-based architectures. It is fundamental to understand reasons why collaborative group-based architectures are better in terms of efficiency and performance for WSN than regular architectures. Fourth section shows that a reduction of the number of transmissions in WSNs (given because of collaborative group-based network) will be translated in a reduction of energy, and consequently an increase of the network lifetime. So, the number of messages needed for collaborative group-based networks is studied using the graph theory in this section. Then, several simulations from our collaborative group-based architecture proposed appear in the fifth section. The sixth section describes an intragroup communication algorithm in order to improve the energy efficiency too and finally, the way to take the most efficient collaborative decisions is presented in the last section.

6.2 Regular Architectures vs. Collaborative Group-based Architectures for WSN

In this section we review the main differences between regular WSNs and collaborative group-based WSNs. First of all, we ought to know the differences between regular and passive WSNs.

- Regular WSNs are networks where all nodes have the same function from the point of view of the network level. Each node measures a parameter, then it is processed, and finally the information is sent to its neighbors. Its neighbors will send the sensed information to their neighbors in order to reach the base station or the sink.
- Passive WSNs are those ones whose sensors sense the environment and send the information to a sink without taking any other action. In an active WSN, when an event occurs, it is notified to the sink and/or to the manage center (MC). Then, the MC sends the necessary information to all nodes (or to some selected nodes) of the network in order to take the appropriate action (sends the information to other nodes, sense more variables, etc.).

Currently passive WSN are not useful because in many environments an intelligent WSN is required. Sometimes, when an event occurs inside the WSN certain tasks should be carried out to perform a specific reply. For example, if we build a WSN to detect fire, the sensors should be able to sense different variables in order to verify the fire and even to monitor it (temperature, CO2, humidity, wind direction, etc.). In this case, sensors may collect these variables and, after some data processing, send the necessary information to a higher processing capacity node in order to activate the appropriate firefighting mechanisms.

Figure 6-1 shows the differences between passive and active WSNs. Red arrows indicate the messages from sensor nodes to the MC and the blue arrows indicate the communication from the MC to the sensor nodes in order to take the appropriate actions. Blue arrows only exist in active WSNs.

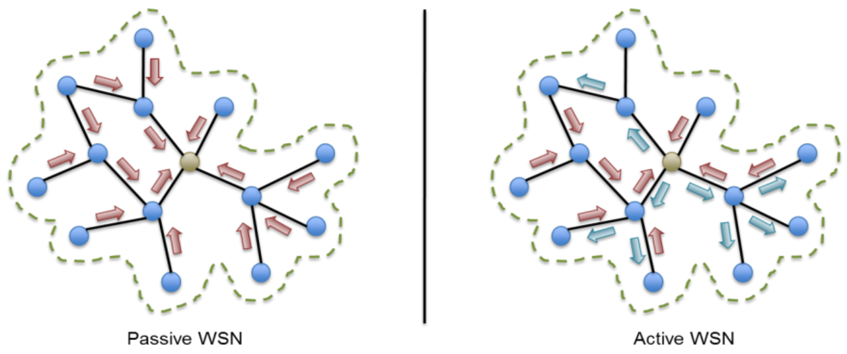


Figure 6-1: Passive vs. Active WSN.

Collaborative group-based sensor networks do not work in the same way. First, they are group-based networks, so the network is logically divided into several small networks (groups). A group is defined as a small number of interdependent nodes with complementary operations that interact in order to share resources or computation time, or to acquire content or data and produce joint results. In a wireless group-based architecture, a group consists of a set of nodes that are close to each other (in terms of geographical location, coverage area or round trip time) and neighboring groups could be connected if a node of a group is close to a node of another group. In [15], we can see that the group-based WSNs provide many benefits. Moreover, cluster-based networks could be considered as a subset of the group-based networks, because every cluster could be a group [20], [99]. But, a group-based network is capable of having any type of topology inside the group, not only clusters. Furthermore, in a group-based network, each group could use a different type of routing protocol. A cluster-based network is made of a Cluster Head (CH), gateways, and Cluster Members (CM). In a cluster, CH nodes fully control the cluster, while in group-based networks, no node controls the group.

Collaborative group-based sensor networks imply cooperation between nodes from the same group and cooperation between nodes from different groups. Thus, the information may be shared only between the most appropriate nodes (from the same groups or from different groups). An event sensed by a sensor implies the exchange of information between different nodes in order to take a final decision and take action in the same place or in another location of the network. Moreover, in a collaborative WSN only parameters that affect the final decision are considered, i.e. the information coming from the neighboring groups of the group that generated the alert. Following the example of the WSN deployment for fire detection, when a sensor detects a fire, the alert message is sent to all nodes in its group, and, after processing the information, and taking into account other parameters such as wind direction, the alert is sent to the affected neighboring groups. Finally, only affected neighboring groups will perform the appropriate actions that other groups will not have to do.

From the point of view of the network messages and energy consumed, in a regular WSN when a node registers an alert, it transmits the alert to all its neighbors, these neighbors transmits the alert to their neighbors and so on. So, the alert is spread to the entire WSN without control. Moreover, a node could receive several times the alert message from different neighbors (see **Figure 6-2**). This situation leads to excessive energy consumption. A collaborative group-based WSN is built based on defined areas or as a function of the nodes' features. Moreover, each group is formed by nodes that interact to share resources or to acquire data to produce joint results [21]. In this case, when a sensor detects a new event, this sensor sends the information to all the members of the group and, depending on the case, the neighboring groups could share this information in order to reach all sensors of the WSN or just some groups. Only the closest sensors to the edge of the group will transmit the information to the sensors of other groups (see **Figure 6-2**). This fact avoids raising considerably the global energy consumption of the WSN, which is very important to enlarge the lifetime of the WSN.

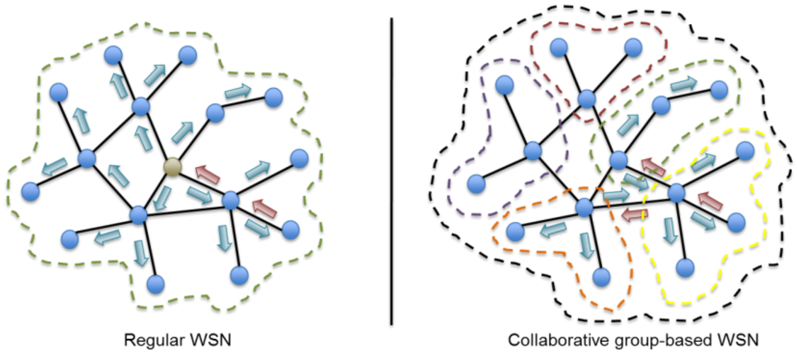


Figure 6-2: Regular WSN vs. Collaborative group-based WSN.

Although we are mainly talking about collaborative group-based WSN with fixed sensors, they could be mobile sensors. In our proposal we consider group-based WSNs where mobile sensors move only inside the boundaries of the groups. Failures could happen in each group. In this case, the mobility will affect in the same manner as in a non-group-based WSN.

In [21] we presented an example of collaborative group-based sensor network. The WSN formation is performed in the same manner of a group-based network but introducing cooperation issues. In addition, each group selects the best connection between sensor nodes taking into account the proximity and the nodes' capacity [13]. In order to have an efficient group-based wireless sensor network, the groups have to communicate with their neighboring groups. When a node detects an event, it warns the alert, jointly with the parameters measured, to the nodes of its groups and, routing the information, to its neighboring groups (not to all groups) based on the location of each group or any other parameter. The location of the sensors could be entered manually or using GPS [73], and a position-based routing algorithm [100] could be used to send the message to the appropriate situation. Neighboring groups could reply to the group that firstly sent the alert if any of the parameters that caused the alert is changed, in order to take the appropriate actions. Cooperation with other groups could change the direction of the alert propagation and the level of the alert.

Figure 6-3 compares a group-based topology and a collaborative group-based topology. In a group-based WSN, all groups will be aware of an event occurred inside the Group 1. But, the network efficiency is higher than in a regular topology [58] because are not transmitted so many messages among all nodes, only communications between groups are considered through end nodes of each group. So, it is a good step forward in terms of efficiency and improved performance of these networks. However, in collaborative group-based networks this efficiency is still greater, because the decision is taken based on the information shared and only the neighboring groups are aware of it. The other groups could be in sleep mode. Sleep groups mean saving energy and not transmit unnecessary information inside the network.

As you can see in **Figure 6-3** the neighboring collaborative groups of group 1 are group 2 and 5. So, in this case they are the physical neighbors and only they are aware of an event occurred inside the group 1. Only the necessary information is transmitted to group 2 and 5 through the only one end node in group 1.

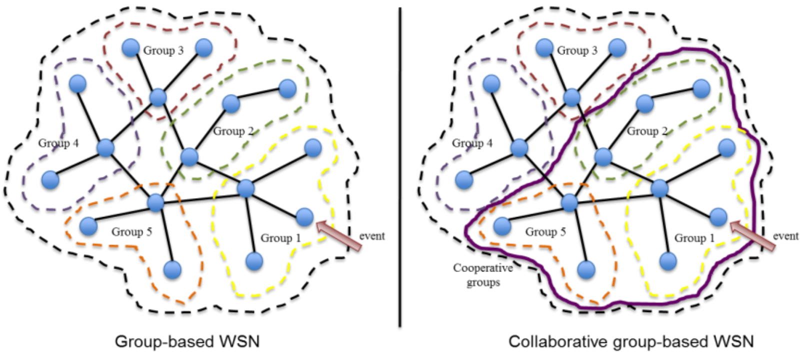


Figure 6-3: Group-based vs. Collaborative group-based sensor networks.

Up to this point, we have analyzed the different types of topologies for WSN and the best architecture has been highlighted. So, now we are going to explain a simple example [21] that describes exactly the collaborative group-based sensor network operation. In **Figure 6-4** shows a group-based topology example. In a group-based network, all groups will be aware of an event produced inside the Group 1. The network efficiency would be yet higher than in a regular topology [2]. But, in cooperative group-based networks this efficient is greater, because only the neighboring groups will be aware of it. The other groups could be in sleep mode. The sleep groups will be saving energy and they would not transmit unnecessary information.

For instance, we define 3 alert levels. *Level_1* means the maximum alert level and *Level_3* the lowest. This collaborative group-based architecture runs following the next steps:

1. A sensor detects a warning event.
2. It senses other parameters (it could be done using a multi-sensor node).
3. The node assigns an alert level for each neighboring group and routes the information for each group using a position-based routing algorithm.
4. The destination group senses the same parameters when it receives the information
5. If the parameters are close, the neighboring groups will accept the alert level, if they are not close; they change the level according to the measured parameters and send the new parameters to the source group.
6. When the source group receives the information, it analyzes the level that should give to the neighboring groups and sends them the new level. Then go to step 4.

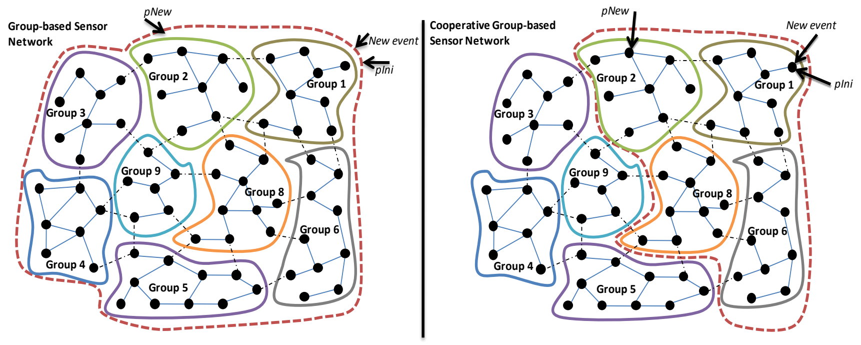


Figure 6-4: An example of group-based and collaborative group-based sensor networks.

Figure 6-5 shows the flow chart of the steps explained.

Taking into account **Figure 6-4** (right side), the messages sent are shown in **Figure 6-6**. Let's imagine that a plague of insects produces an event in Group 1, and the other parameters sensed are the wind direction and speed (now called pIni). *Node*$_i$ calculates the level for each group and sends the alert and pIni to all nodes in its group. Each message will be also routed to the appropriate neighboring groups as in [20]. The information reaches Group 2, Group 8 and Group 6. Let's suppose that the wind comes from east, so Group 2 has *level_1*, Group 8 has *level_2* and the Group 6 has *level_3*.

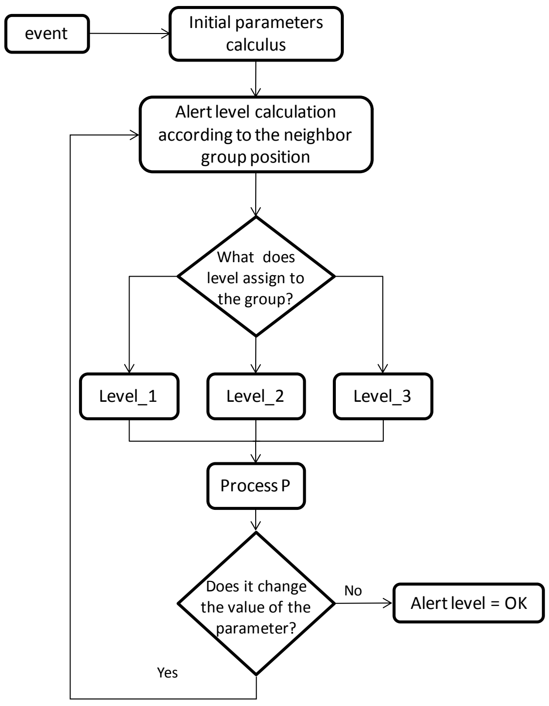

event
Initial parameters calculus
Alert level calculation according to the neighbor group position
What does level assign to the group?
Level_1
Level_2
Level_3
Process P
Does it change the value of the parameter?
No
Alert level = OK
Yes

Figure 6-5: Collaborative architecture operation.

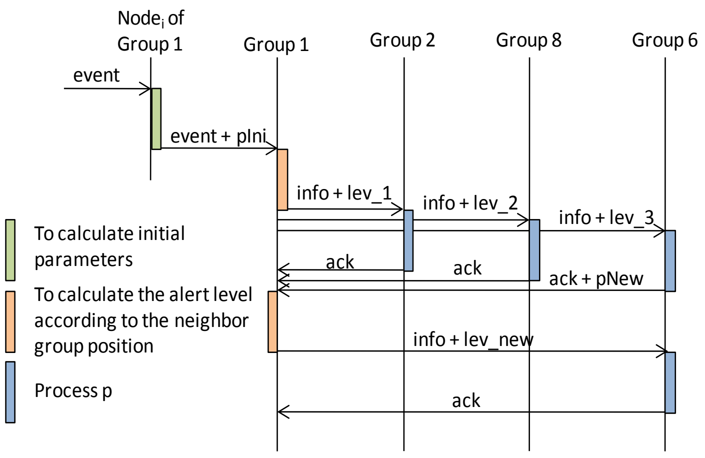


Figure 6-6: Messages exchanged.

```
If  (pNew  ≤  pIni*1.1  ||  pNew  ≥
pIni*0.9)
          Send ack to source group
          Alert level assigned = OK
Else
          Send ack + pNew

End
```

Figure 6-7: Process p operation.

The p process is executed in each group when the information arrives. **Figure 6-7** shows a simple algorithm. If the parameters sensed are close to the initial parameters, the alert level does not vary. Let's suppose that Group 6 detects changes in the wind direction (pNew). This group will send the new wind direction to Group 1, and then this group will estimate the new alert level.

So, when a sensor detects a new event, an alert is sent to its group and it is distributed to neighboring groups. The collaboration between groups is used to send messages between groups in order to obtain the right alert level. This system is going to be developed as a fire detection system where the wind, humidity and temperature parameters are also measured. There are other application environments such as Rural and agricultural monitoring and natural crisis.

6.3 Energy Analysis in a Collaborative Group-based WSN

This section analyzes the energy needed to transmit packets in a collaborative group-based WSNs architecture and it is compared with a regular WSN architecture. In order to do that we perform an important mathematical analysis to show that group-based topologies imply a significant energy saving and they are the most efficient mode for WSN.

The notation used in this analysis and its definition is shown in **Table 6-1**. φ_{11}, φ_{12} and φ_2 are constant radio parameters, typical values are φ_{11}=50nJ/bit, φ_{12}=50 nJ/bit, φ_2=10 pJ/bit/m^2 (when n=2) or 0.0013pJ/bit/ m^4 (when n=4).

We follow the model presented in [101]. The energy consumed by a sensor to transmit and receive a data packet between two nodes is given by Eq. 6-1.

$$E = (\varphi_{11} + \varphi_{12}) \cdot r + \varphi_2 \cdot (d^n) \cdot r \qquad \text{Eq. 6-1}$$

Where d is the distance between both nodes. Let be D the distance between the sending node of a group and the closest node from other group. Thus, $E^{opt}(D) \geq E^{opt}(D)$, being $E^{opt}(D)$ the minimum power to transmit a data packet from the node to the other group. $E^{opt}(D)$is equal to Eq. 6-2 if and only if D is multiple integer of $d_{opt} = \sqrt[n]{\frac{\varphi_1}{\varphi_2 \cdot (n-1)}}$, as we can see in [34].

Table 6-1: Notation and definition.

Parameter	***Definition***	***Parameter***	***Definition***
φ_{11}	Power required to turn on the transmitter	r	Bit rate (bits per second)
φ_{12}	Power required to turn on the transmitter	s	Number of sensors in the WSN
φ_1	$\varphi_{11} + \varphi_{12}$	R	Average area radius of a WSN
φ_2	Power to transmit	S_i	Number of sensors in the group i
d	Distance between 2 communicating sensors	R_i	Average area radius of the group i
d_{opt}	Optimum distance between 2 sensors	J_m	Number of collaborative groups
n	Path loss exponent (typical values: 2 or 4)	J	Number of groups, m Є [1,J]

$$E^{opt}(D) = \left(\varphi_1 \cdot \frac{n}{n-1} \cdot \frac{D}{d_{opt}} - \varphi_{12}\right) \cdot r \qquad \text{Eq. 6-2}$$

$E^{opt}(D)$is the lower bound of energy consumption in the flat scheme without data aggregation, which indicates an ideal case where the per-hop distance for transmission is *d_{opt}* meters. According to the energy model in [102] and the previous assumptions, the expected energy consumption per second of a group is given by Eq. 6-3.

$$E = (S_i - 1) \cdot \left(\varphi_1 + \varphi_2 \cdot \frac{2 \cdot R_i^n}{n+2}\right) \cdot r + (\varphi_{11} + \varphi_2 \cdot D^n) \cdot r \qquad \text{Eq. 6-3}$$

It is assumed that there are *J* groups in the network and each regular node only needs to transmit its data packet to the head node or to the end node of its group. The average radius of each m group can be regarded as $R/\sqrt{J}_m$, and the expected energy consumption per second in all regular nodes is given by Eq. 6-4.

$$E_{rn} = (s - J_m) \cdot \left[\varphi_{11} + \varphi_2 \cdot \frac{2}{n+2} \cdot \left(\frac{R}{\sqrt{J}}\right)^n\right] \cdot r \qquad \text{Eq. 6-4}$$

The energy consumption for all end nodes is given by Eq. 6-5.

$$E_{bor} = (s - J_m) \cdot \varphi_{12} \cdot r + J_m \cdot \left[\varphi_{11} + \varphi_2 \cdot \left(\frac{R}{\sqrt{J}}\right)^n\right] \cdot r \qquad \text{Eq. 6-5}$$

In order to evaluate the energy consumed in a network without collaborative groups, we consider Eq. 6-4 and Eq. 6-5 for a single group (*J*=1), and only one sink node. Then, Eq. 6-6 and Eq. 6-7 are obtained respectively.

$$E_{rn} = s \cdot \left[\varphi_{11} + \varphi_2 \cdot \frac{2}{n+2} \cdot (R)^n\right] \cdot r \qquad \text{Eq. 6-6}$$

$$E_{bor} = [s \cdot \varphi_{12} + \varphi_{11} + \varphi_2 \cdot R^n] \cdot r \qquad \text{Eq. 6-7}$$

The global energy consumption per process E_{proc} (transmit or receive a data packet) is given by equation Eq. 6-8 both for collaborative networks and for networks without collaborative groups.

$$E_{proc} = E_{rn} + E_{bor} \qquad \text{Eq. 6-8}$$

6.3.1 Analytical Comparison

In order to observe the energy-saving improvements provided by a collaborative group-based WSN compared to regular WSN, we have taken into account Eq. 6-8 for both cases (regular and collaborative group-based WSN). Following the typical values of the constants that appear at the beginning of this section (φ_{11}=50 nJ/bit, φ_{12}= 50 nJ/bit, φ_2 = 10 pJ/bit/m^2), n=2 is the most appropriate value for outdoor communications. Furthermore, m=2 has been chosen, network has 100 nodes (s=100) and the network radius equals to 200 meters (R=200). The above equations have been programmed and varying the number of groups and bitrate, we obtain **Figure 6-8**. It shows the global energy consumption for different values of Jm. When J_m=1, we talk about just one group (the whole WSN), so we can see that the energy consumption is higher than in any other value of J_m (higher values of J_m imply more groups). By increasing J_m we see that consumption decreases, but each time the difference of slope is lower.

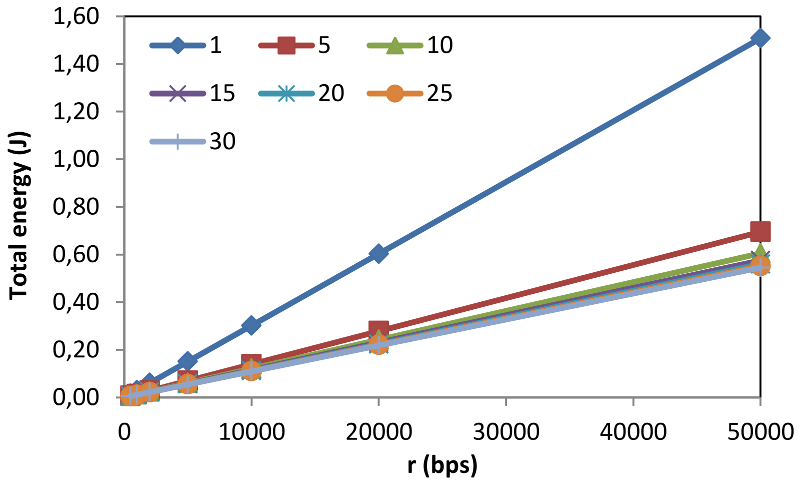


Figure 6-8: Energy of the collaborative group-based WSN.

6.4 Efficient Communications in a Collaborative Group-based WSN

In our collaborative group-based WSN from the point of view of network level all nodes are alike and they may be mobile. There are not base stations, sinks or management nodes. For this reason, all nodes can make decisions of their connections according to the algorithm explained in section 5.2. Moreover, all communications between the sensor nodes are done by wireless links.

As usual, we can model a WSN using the graph theory, where a WSN is defined as $G = (S, E)$, where S is the set of sensor nodes ($|S| = s$) and $e = (i, j) \in E$ represents a wireless link between the nodes *i* and *j* only if they are in their communications range.

Any two nodes that are connected by an edge are neighbors to each other. Nodes in *G* can send (and receive) messages to (from) their neighbors. Every node $v \in S$ has a unique identity *ID(v)*. Each node initially knows its own identity and the identities of its neighbors in one jump inside *G*.

The size of a group is the number of nodes belonging to it. According to the definition of our WSN, there are three types of nodes: head, leaf and end nodes. For example, the head node of the group *k* is defined as $C(k)$. The number of nodes in a group is *B*, where *B* is smaller than *s*.

Other parameters that are used for defining communications in sensor networks is the distance between two nodes, $d(i, j)$, it is the Euclidean distance between the nodes *i* and *j*. As it is a group-based network, each group will be defined as $G(k)$, in this case this is the group *k*.

From the graph theory textbook [103] we will denote $\Gamma_k(i)$ the k-neighborhood of a node *i*, e.g., $\Gamma_k(i) = \{v \in S | 0 < d(i,j) \leq k\}$ and we will denote $\delta_k(i) = |\Gamma_k(i)|$. Besides, we will denote $e(i/G) = \max_{v \in G(i)}(d(i,j))$ the eccentricity of a node *i* inside its group. Thus the diameter of a group will be $D(G(i)) = \max_{v \in G(i)}(e(v/G))$.

Once the definition of a group-based WSN has been done, we analyze the number of messages sent inside the network when an event occurs. Then, this information has to be sent to other nodes and this information has to be sent to other nodes. According to [36], in the worst situation, the number of messages in a regular network when there is an alert is $O(s)$, while in a collaborative group-based network the number of messages is $O((m+1)B^2)$, where *m* is the average number of collaborative groups in the network. If $B << s$ we can affirm that the number of messages in a group-based network will be lower than in regular networks, for this reason $O((m+1)B^2) << O(s)$.

6.4.1 Analytical Comparison

In order to validate this analysis, we perform a simulation where there is a WSN of *s* nodes. These nodes are distributed in a square area of length *l* units. The nodes are randomly placed. The average density of nodes per unit length is s/l^2. In this model, the nodes have a link with another node if and only if they are within a distance *d* units of each other. Consequently, a node will have an average of $(\pi d^2 s/l^2) - 1$ neighbors. This leads to an edge probability of $(\pi d^2 s/l^2 - 1)/(s-1)$ which is the probability that two nodes chosen at random in the network are connected by a link. In particular, for a large value of *s*, the edge probability is approximately $\pi d^2/l^2$.

Table 2 shows the average number of end nodes, the average number of groups, the likelihood of end nodes, the number of messages that will be in the case of *l* = 25 units of length and *d* = 1 unit and the percentage of improvement respect to a regular architecture (without groups). The observed data in this table have been obtained using the previous boot parameters and applying the formulas presented in this section. We have used a mathematical program where the formulas have been programmed and then we have introduced the different starting values for obtaining these results (see Table 2). We have estimated all these parameters when *B* varies between 1, 2, 4, 8 and 16, with an average number of collaborative groups (*m*) ranging from 0 to 5 (we have selected 0, 1, 3 and 5), and when the number of nodes in the whole WSN varies between 250, 500 and 1000.

Table 6-2: Results of the communication analysis.

	s	***Average nº of end nodes***	***Average nº of groups***	***Likelihood of end nodes***	***Nº of messages***	***% of improvement***
B=1, m=0	250	0.256	-	0.0001028	250	-
	500	1.512	-	0.00303	500	-
	1000	4.024	-	0.00403	1000	-
B=2, m=1	250	0.256	125	0.0001028	8	96.8 %
	500	1.512	250	0.00303	8	98.4 %
	1000	4.024	500	0.00403	8	99.2 %
B=4, m=3	250	0.256	62.5	0.0001028	64	74.4 %
	500	1.512	125	0.00303	64	87.2 %
	1000	4.024	250	0.00403	64	93.6 %
B=8, m=3	250	0.256	31.25	0.0001028	256	No improvement
	500	1.512	62.5	0.00303	256	48.8 %
	1000	4.024	125	0.00403	256	74.4 %
B=16, m=5	250	0.256	15.625	0.0001028	1536	No improvement
	500	1.512	31.25	0.00303	1536	No improvement
	1000	4.024	62.5	0.00403	1536	No improvement

According to the results shown in **Table 6-2**, we can conclude that as the number of nodes per group (B) is increasing, the efficiency in terms is increasing too. But, it is not so if the number of nodes in the whole network (s) is not considerably higher than the number of nodes per group (B). So, this proposal is a good solution for a large WSN with an appropriate number of nodes per group. It is so because when the groups are large ($B\uparrow\uparrow$) compared with the whole network, the WSN starts to be as a regular WSN. The architecture needs more control messages to manage the group-based architecture. So, it is necessary a tradeoff between the number of network nodes and the appropriate number of groups to improve the performance of a WSN with a collaborative group-based architecture.

Figure 6-9 shows the maximum number of broadcasting messages needed when we have several sizes of groups and several average collaborative groups. In this case the number of nodes *s* was 1000 and *l* was 25 units. In this figure we can observe that increasing the group size (*B*) and the number of collaborative groups, it increases the number of messages in the WSN. The simulation shows that when a high number of messages is expected in the WSN, it is better a group-based WSN with a low *B* and with few collaborative groups. This happens with the broadcasting messages, but if we analyze the management messages, when we have small groups there are a lot control messages to form and manage the groups. For this reason, a compromise between the group size, the number of collaborative groups and the number of broadcasting messages should be reached. Seeing the **Figure 6-9**, the best solution could be a group size between 4 and 6 nodes and an average number of collaborative groups between 3 and 5.

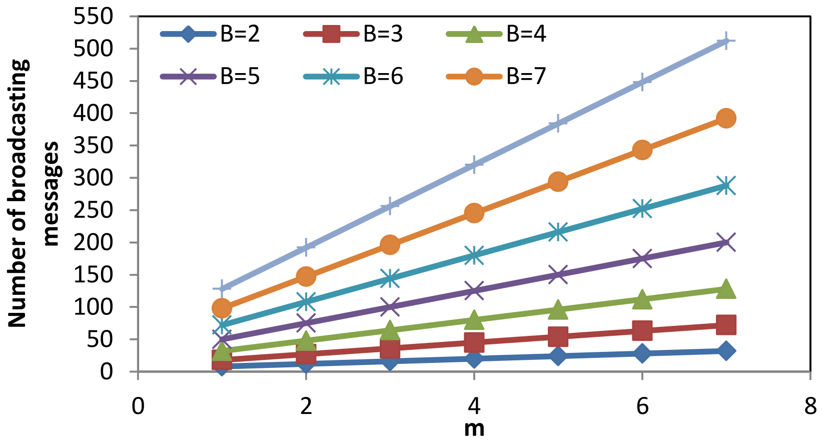


Figure 6-9: Maximum number of broadcasting messages for different group size and varying the average number of collaborative groups.

6.4.2 Collaborative Group-based Network Simulation

In order to evaluate the system proposed in this chapter. We have simulated several scenarios using the OPNET Modeler network simulator [83]. In next simulations we are going to see the behavior of the collaborative group-based WSN according to the number of groups in network. The test scenario has 100 sensor nodes placed in 500 × 500 meters. We have increased the number of groups in each simulation. We have chosen DSR as routing protocol because in the Chapter 3 we can see that this protocol has a good behavior. It is not the best nor the worst. Instead of a standard structure we have chosen a random topology. The nodes can move randomly during the simulation. The physical topology does not follow any known pattern. The obtained data neither depend on the initial topology of the nodes nor on their movement pattern because all of it has been fortuitous. The sensor nodes have a 40 MHz processor, 512 KB RAM memory, a radio channel bandwidth of 1 Mbps and their working frequency is 2.4 GHz. Their maximum coverage radius is 50 meters. This is a conservative value because usually the nodes in sensor networks have larger coverage radius, but we preferred to have lower transmitting power for the sensor devices in order to enlarge their lifetime.

The traffic load used in the simulations is MANET traffic generated by OPNET. We inject this traffic 100 seconds after the beginning. The traffic follows a Poisson distribution (for the arrivals) with a mean time between arrivals of 30 seconds. The packet size follows an exponential distribution with a mean value of 1024 bits. The injected traffic has a random destination address, obtaining a simulation that does not depend on the traffic direction. In **Figure 6-10** we see that the traffic injected into the simulation follows the same pattern for all scenarios. The average traffic is 100 Kbps (an adequate traffic for WSNs).

Figure 6-11 and **Figure 6-12** show the total traffic in the WSN. In **Figure 6-11**, the total traffic in the network is too high compared when the topology that uses collaborative groups. When the network does not have collaborative groups, the average total traffic is around 6500 Kbps, when we have 2 collaborative groups it decreases 95%. In **Figure 6-12**, we only see the collaborative group-based topologies to show better the results. The total traffic decreases when the number of groups increases. As we can see in **Figure 6-12**, when we have 2 groups, the average total traffic is around 310 Kbps, but when we have more groups (e.g. 6 groups), the total traffic decreases down to 140 Kbps. This demonstrates that collaborative group-based WSNs have lower traffic.

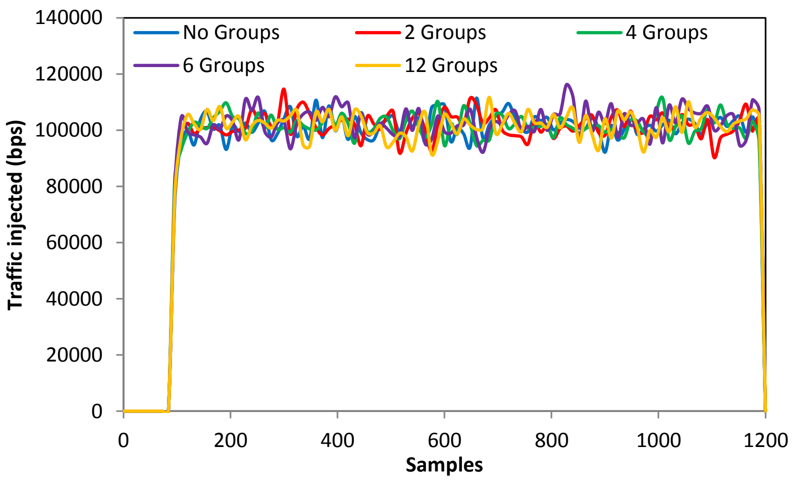


Figure 6-10: Traffic injected in each scenario.

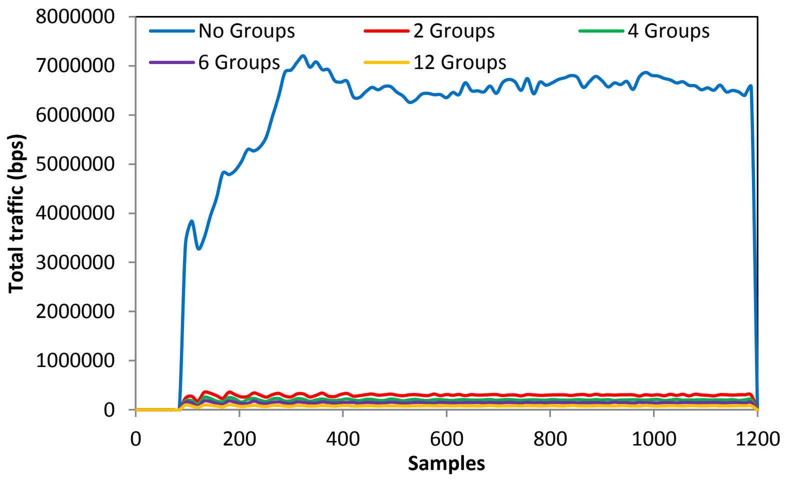


Figure 6-11: Total traffic in each scenario.

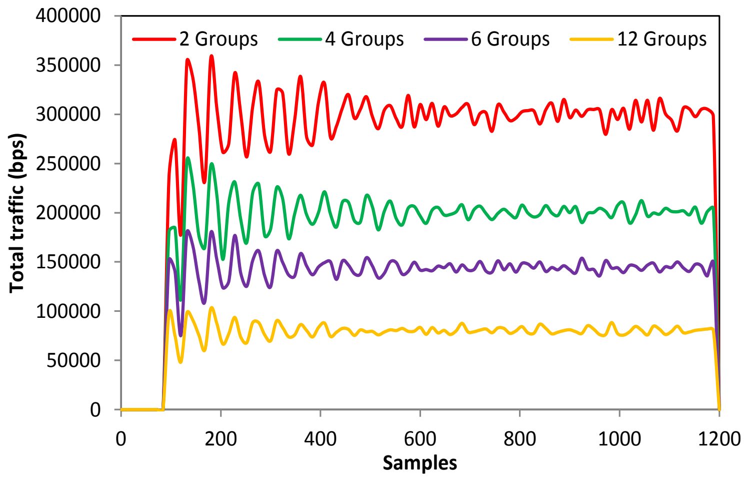


Figure 6-12: Total traffic in collaborative group-based scenarios.

Figure 6-13 and **Figure 6-14** show the delay in our network. In **Figure 6-13**, we can see that the difference between collaborative group-based topologies and a regular network is very high. In a regular WSN, the delay is around 120 seconds (which is too high for any application). This high value is obtained because the mobility of nodes. Constant mobility causes that the nodes need to compute their routing table constantly, and this process takes more time to arrive at the convergence state. This delay could change according to the used routing protocol. Anyway, its difference with collaborative group-based WSN is high.

In **Figure 6-14** we can see the delay when we apply collaborative groups in the WSN. In all cases the delay is lower than 0.05 seconds. There is a peak at around 100 samples, because in this moment the simulation starts. Then, when the WSN converges the delay is lower and very constant. There is quite difference between collaborative group-based WSN and regular WSN. When there are not collaborative groups, the nodes are not segmented, and, for this reason, we need more resources to manage the network. When the WSN is divided into collaborative groups, the management process is also divided, for this reason we need less resources.

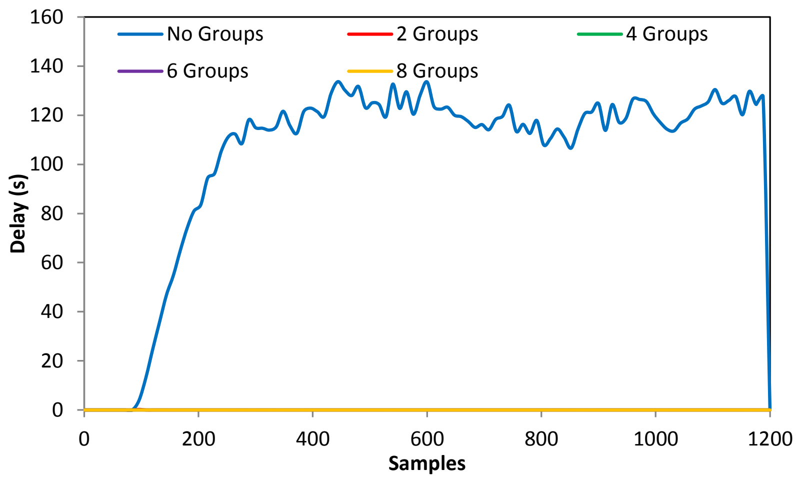


Figure 6-13: Delay in each scenario.

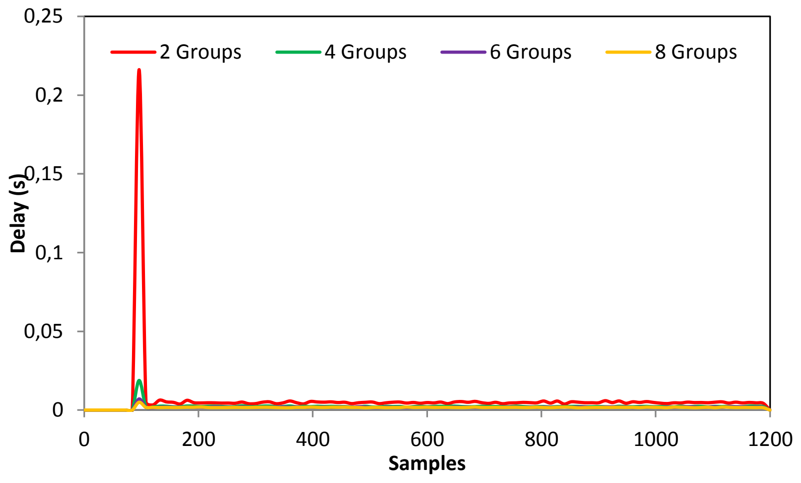


Figure 6-14: Delay in each collaborative group-based scenario.

In **Figure 6-15** we show the average number of hops needed to arrive to a destination when we are using groups and when we are not. As we can see, when we have a regular WSN, the average number of hops is around 6, so we need to cross 6 nodes to arrive to the destination. When we have collaborative groups in the network, the average number of hops is the half. We need 3 hops to arrive to the destination when we use collaborative group-based topologies.

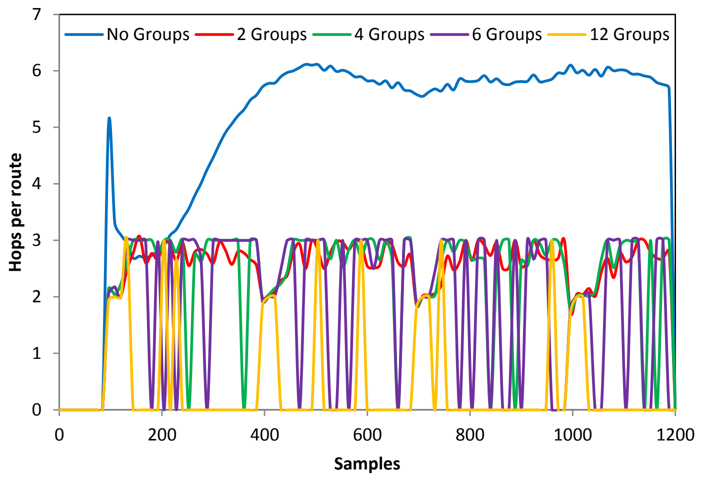


Figure 6-15: Hops per route needed to arrive at the destination.

In **Figure 6-16** and **Figure 6-17** we analyze the number of ACKs sent in regular and group-based WSNs. In these cases we observe the same behavior as in previous figures. The regular architecture needs more ACKs (2400) than the collaborative group-based architectures (see **Figure 6-16**). This is because regular WSNs need more messages to manage the architecture.

In order to see better the total number of ACKs sent in collaborative group-based WSNs, we show **Figure 6-17**. In this figure we can see that the number of ACKs sent follow the same pattern for all topologies independently of the number of groups, although each topology inserts more or less ACKs. When we have 2 groups, where each group manages 50 sensor nodes, the average total ACKs sent is around 400. This number drops to half when the number of groups is equal to 4. But, **Figure 6-17** shows that although we increase the number of groups, the total ACKs sent will not be less than a certain value. In this case, for a topology with 6 or 12 groups, the total number of ACK sent approximately equals 75.

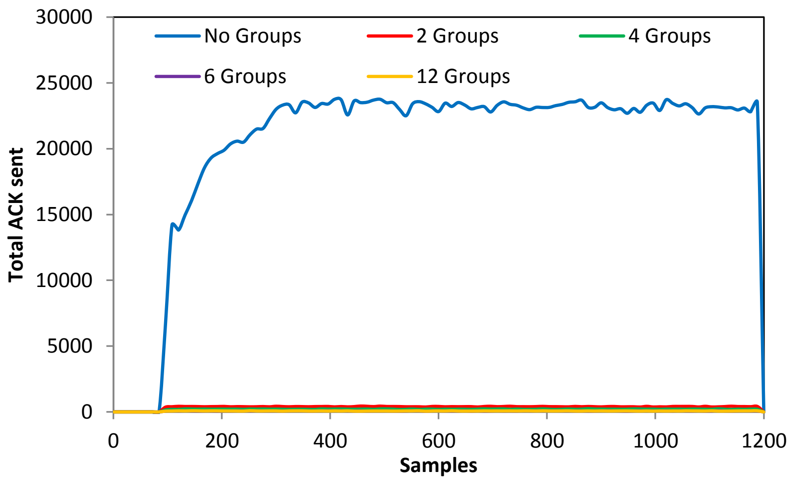


Figure 6-16: Total ACK sent in each scenario.

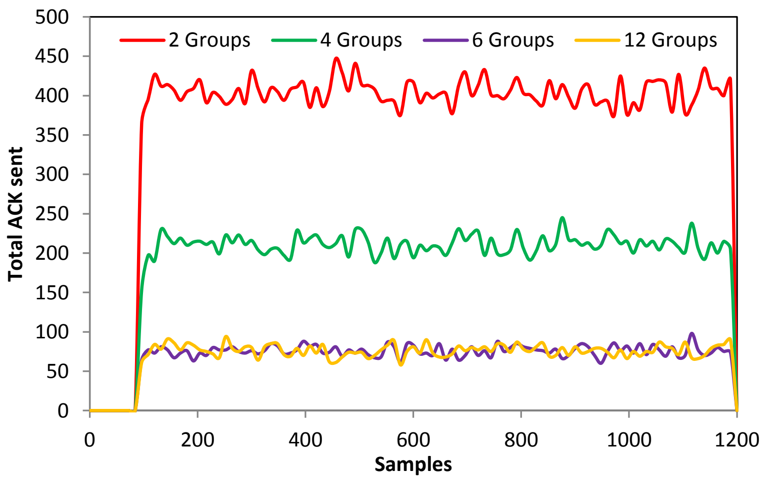


Figure 6-17: Total ACK sent in collaborative group-based scenarios.

Figure 6-18 and **Figure 6-19** show the retransmission attempts for all cases abovementioned. In **Figure 6-18** we see that the regular WSN needs approximately 1.5 retransmission packets to guarantee the correct running of the system. But this retransmission is not needed when our system is based on collaborative groups, because the required management is done by collaborative group-based WSN.

When we increase the number of groups we need less number of retransmissions. When we have two groups, the average number of retransmissions is less than 0.1 packets, so it is negligible (see **Figure 6-19**). We can notice that when the number of groups is less than 4, the retransmission packets could be zero. Analyzing these simulations (**Figure 6-18** and **Figure 6-19**) we can affirm that if we use collaborative groups in our topologies we need less retransmissions, even they could be zero depending on the system.

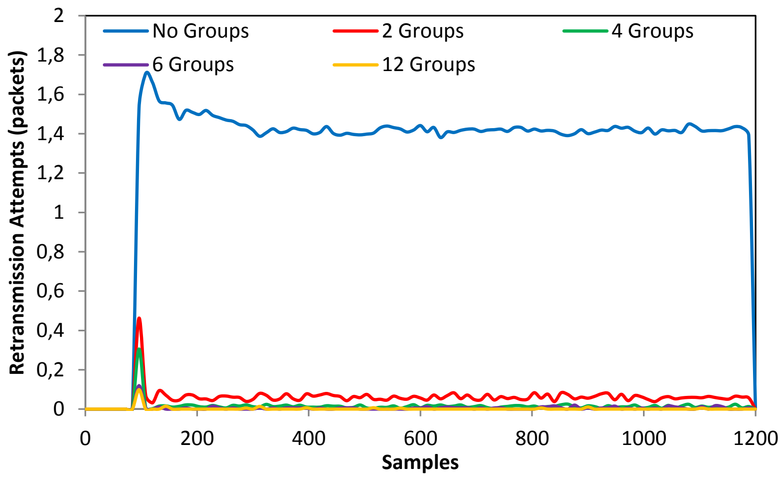


Figure 6-18: Retransmission Attempts in each scenario.

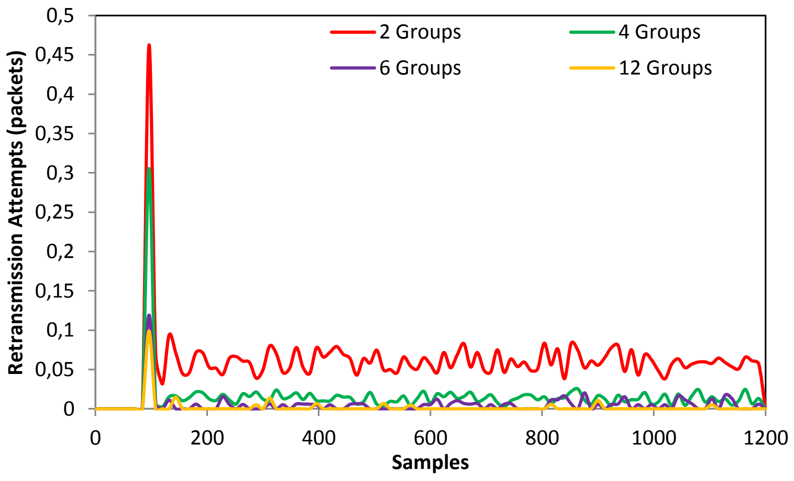


Figure 6-19: Retransmission Attempts in collaborative group-based scenarios.

Finally, in **Figure 6-20**, we present the load processed by a collaborative group. In this figure we only focus on the group-based WSNs, because, as we have seen in the previous figures, the regular WSN has worse performance. In this figure we see that when we have 2 groups in our network, the load is around 150 Kbps, this load decreases down to 60 Kbps when we have 4 groups, 25 Kbps for 6 groups, and less than 10 kbps for 12 groups. This happens because when we have more collaborative groups, the number of nodes managed per group is lower. When we have a lot of collaborative groups in the WSN, we need more control information to manage it correctly. For this reason, when we select the number of groups, we should think several issues: to take into account the efficiency at network level, and take care of the management information needed to create and manage each collaborative group.

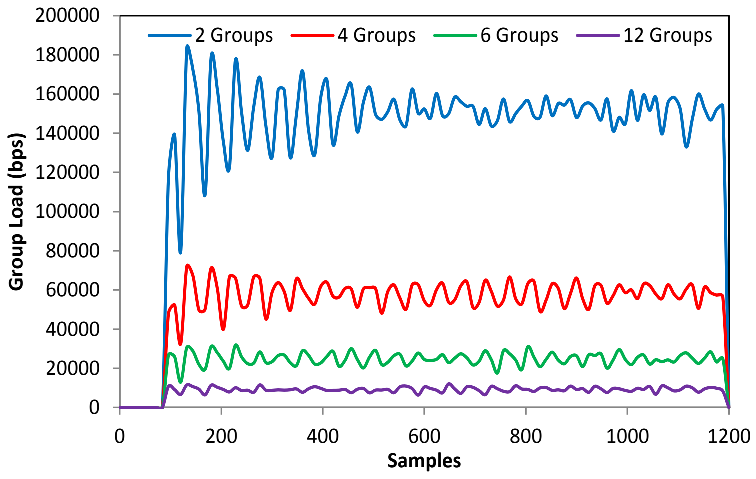


Figure 6-20: Load processed by a group.

6.5 Intergroup Communications in Group-based Wireless Sensor Networks

In this section, we show a method of collaboration between groups based on the communication between neighboring groups. Using this information, the groups will be able to create a collaborative intergroup routing pool tree. This routing pool tree allows collaboration between neighboring groups and also efficient delivery of data from any group to the sink node. This collaborative group-based system has been simulated and we have shown that it increases significantly the lifetime of sensor nodes.

A collaborative group-based WSN is built on defined areas or as a function of the nodes' features. Moreover, each group is composed of nodes which interact to share resources or to acquire data to produce joint results as it is shown in **Figure 6-21**. The network is divided into logical groups which include an intragroup routing protocol. This intragroup routing protocol could be any of those used in WSN.

When a sensor detects a new event, this sensor sends the information to the rest of members inside the group using its intragroup routing protocol. Depending on the case, the neighboring groups can share this information in order to reach all sensors of the WSN or just some groups. Relevant information is shared between sensor nodes from the same group and the most suitable neighboring groups as needed.

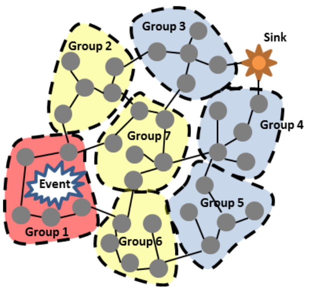


Figure 6-21: Collaborative group-based WSN.

The focus is that only the closest sensors to the edge of the group will transmit the information to the sensors of other groups. This fact reduces considerably the global energy consumption of the WSN. Besides, each group works with an alarm level and each alarm level supposes different energy consumptions in nodes of a group. It means reducing the exchange of information when an event is detected in a group. So, energy consumption to take a final decision and produce an action is minimized and the WSN's lifetime is enlarged. In the second section, we describe exactly the collaborative group-based sensor network operation. It is based on the use of different alarm levels to define which level of danger or importance has an event. In this section we explain that in detail and show the process of creating of the collaborative intergroup communication in a group-based WSN.

6.5.1 Intergroup Communication Algorithm

All nodes in each group have the same functionality, but according its function can be divided into End Nodes because they are on end between two groups and they are responsible of connection between neighboring groups, Head Node which helps to create each group and it is the logical center of every group and Regular Nodes that are the rest of nodes which make up the group along with the Head Node and End Nodes. The process of creating groups is based on sending messages and according to neighboring nodes' replies and a group maximum radius (R_{group}), which is previously saved into all nodes of network, these groups limit their size. Each one knows its neighboring group using these messages, for more information about group's creation you can refer to previous chapter.

Intergroup communication is done by end nodes of each group. These nodes exchange information about groups. They are responsible for exchanging information between groups to send one type of messages or others. This intergroup communication is based on a collaborative intergroup routing pool tree.

The main idea is based on the spanning tree algorithm, where the root is the sink node. The sink node always has the lowest ID, it sends a message (*Ngroup_discover*) to the neighbor's groups with a TTL equal than $2 \cdot R_{group} - 1$. In this way, the message will not reach beyond the center nodes of each group. Sink node or a Head Node, which receives a *Ngroup_discover* message, sends a new *Ngroup_discover*. This message will be routed, according to routing protocol inside the group (intragroup routing), until it reaches new groups. If this message does not find any destination node (Head Node) using $2 \cdot R_{group} - 1$ hops, it will be deleted to avoid extra messages on groups. After that, each head node, that has received a message to create an intergroup routing pool tree, will send another message to their neighbor's groups through its end nodes.

When a head node sends these messages, it receives replies from groups which are not included on the tree and others which have already been included. When a Head Node receives this message, it is processed. In this process each Head Node checks if *Ngroup_discover* messages provides from a group included on the intergroup routing pool or not. According to this checking it will reply with an *XXXgroup_reply,* where *XXX* is the ID of the group*,* to perform the collaborative intergroup routing pool tree. If the group is not included on the tree, it will be considered as leaf group. This is known by *LeafGroup_reply* message. Otherwise, if a group is included on the tree, it is considered as collaborative group and it sends a *CollaGroup_reply*. We can see this algorithm on **Figure 6-22**.

Head node of each group is responsible of selecting if its group is a leaf group of the intergroup routing pool tree or only a collaborative group according to a criterion. **Figure 6-23** represents the intergroup routing pool tree of the network topology shown in **Figure 6-21**. In this case, the criterion is the lowest ID group, so a group will select to join as leaf group according to ID group. But, the neighboring groups will keep collaborative links to exchange information between groups when they don't want to send that information to the sink. In contrast, when nodes want to send information to the sink, they use the path created in the routing tree, because this is the route known by groups.

Figure 6-21 brings the collaborative group-based WSN considered in this work. It comprises seven groups of five sensor nodes each of them. We consider that an event starts in group 1 and this event can create several types of alarms with different energy consumption according its proximity to the event. Each type of alarm detects a different event, so each type of alarm consumes a certain amount of energy. It depends on tasks done by every group, for instance the group which senses an event, it will active functions with higher energy consumption (e.g. activate sprinkler system and smoke extraction in a fire detection). The neighboring groups, of group which detects the event, will only activate smoke extraction. Thus we want to achieve more efficient energy consumption which is essential in WSN. So, in **Figure 6-21** the highest energy level (El_0) is generated in group 1 (represented in red), a lower point of energy level (El_1) is used for its neighbors groups (in yellow), and the lowest energy level (in blue El_2) for the rest of groups, inside of collaborative group radius (R_{cg}). It defines the maximum expansion of cooperation between neighboring groups from the group that detected the event.

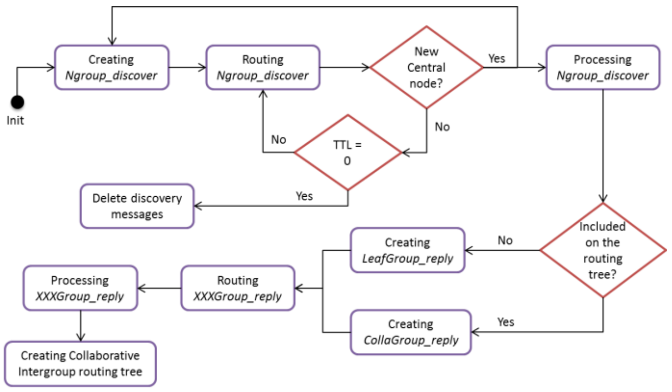

Init
Creating Ngroup_discover
Routing Ngroup_discover
New Central node?
Yes
No
Processing Ngroup_discover
TTL = 0
No
Yes
Delete discovery messages
Included on the routing tree?
No
Yes
Creating LeafGroup_reply
Creating CollaGroup_reply
Routing XXXGroup_reply
Processing XXXGroup_reply
Creating Collaborative Intergroup routing tree

Figure 6-22: Collaborative intergroup routing algorithm.

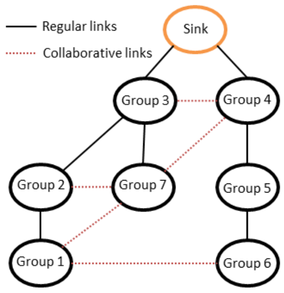


Figure 6-23: Collaborative intergroup routing pool tree.

6.5.2 Simulation and Results

We have simulated different topologies in order to show that energy consumption is really more efficient with our proposal. These simulations have been done using Matlab [14]. Inside each group it is running an easy routing protocol based on flooding concept. The maximum packet size is 100 bytes on the application layer and the event is generated when the group-based WSN converges. We have chosen sixteen sizes of WSNs, from 40 to 200 sensor nodes (SN_s), and 6 procedures of grouping, from 5 to 10 SNs (Sensor Nodes) per group (G_{SN}). In order to calculate the energy required to collaborate between groups (E_{col}), we have used the Eq. 6-9, where NG_i represents the neighboring groups with i distance from the event detecting group. Moreover, El_i denotes energy level of group i.

On the other hand, we have to take into account energy required to receive and transmit data packets between groups (see Energy Analysis in the third section of this chapter). This energy is represented in Eq. 6-10 with the name E_{proc}. Notation used in Eq. 6-10 is represented in **Table 6-1**. Finally, the total energy is E_T (see Eq. 6-11).

$$E_{col} = G_{SN} \cdot \left(El_0 + \sum_{i=1}^{R_{cg}} El_i \cdot NG_i \right) \qquad \text{Eq. 6-9}$$

$$E_{proc} = \left[(s - J_m) \left[\varphi_{11} + \varphi_2 \frac{2}{n+2} \left(\frac{R}{\sqrt{J}} \right)^n \right] + (s - J_m)\varphi_{12} + J_m \left[\varphi_{11} + \varphi_2 \left(\frac{R}{\sqrt{J}} \right)^n \right] \right] r \qquad \text{Eq. 6-10}$$

$$E_T = E_{proc} + E_{col} \qquad \text{Eq. 6-11}$$

In this section we calculate E_{col} required, because E_{proc} has already been studied in the third section. We have considered that one sensor node consumes 3mW in the highest energy level group, 2mW for their neighbors groups and 1.5mW for the rest of groups in a group-based WSN (these energy costs are provided by a real sensor node called Waspmote [15]. In **Figure 6-24**, we can see E_{colla} required by WSNs when they are collaborating between groups and when they are not. All sensor nodes consume 3mW when there are not collaborative groups. Thus **Figure 6-24** shows that as the number of sensor nodes increases, the difference between energy required by a collaborative group-based WSN and a non-collaborative group-based one grows (for example, with 200 nodes the E_{colla} required in around 600 mW when we have non-collaborative group-based network, this energy decreases until 325 mW when we have collaborative groups. Besides, it is important to note that for non-collaborative group-based sensor networks, the energy required increases with a steeper slope than for collaborative group-based networks where the increment is slower and so slower. In other words if our network has 40 sensor nodes, the difference between the energy required when they are organized into collaborative groups compared to when they are not, is 25% lower. In contrast, it is 48% lower for a network composed by 200 sensor nodes. Therefore, as the number of sensors increases in a WSN, the efficiency of organizing sensor nodes into collaborative groups grows. This difference represents the energy saved by collaborative grouping regarding a WSN with the same size but no groups. We can see it in detailed in **Figure 6-24**.

Although it seems the E_{colla} required is similar for all types of grouping in **Figure 6-24**, we have extracted that according to number of sensor nodes per group, we can get a little more energy efficiency. It is at most 10%, so this improvement is insignificant compared to the improvement brought by collaboration. Despite this, we have seen fit to show the percentage of energy saving according to the number of sensor nodes per group in **Figure 6-25**.

If we pay attention to energy saving in **Figure 6-25**, in general we see the smaller groups are more efficient than the largest ones. In our case, the most efficient is 5SNs per group for most network sizes, but the curve of 10SNs per group is very steady as occurs with 5 SNs per group. However, the shortest groups need more management traffic for their correct performance. So, we should choose which the best solution is according to the environment which we are going to monitor by WSNs.

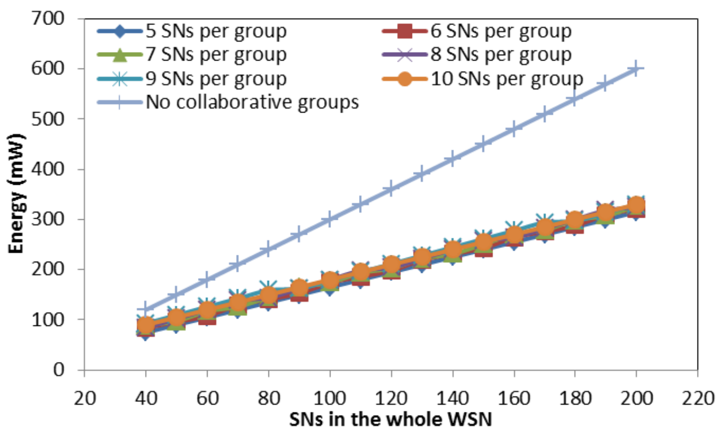


Figure 6-24: E_{colla} required by collaborative group-based WSN.

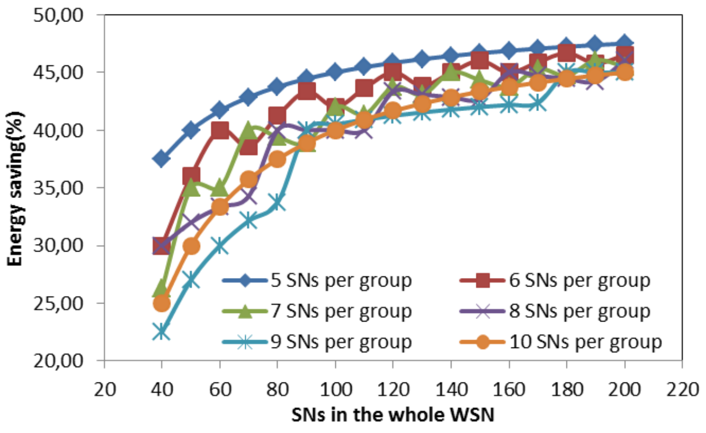


Figure 6-25: Percentage of the E_{colla} saving for each grouping.

6.6 Taking Decisions in a Collaborative Group-based WSN

In this section we explain a collaborative decision system for collaborative group-based WSNs in an environmental monitoring application. In this system the network is divided into groups, in which each group is formed by some sensor nodes which measure several parameters. These parameters and the collaboration between groups make an efficient monitoring network. Each sensor node gathers four parameters: temperature of the environment $T = \{t_0, t_1, t_2, t_3, \dots, t_n\}$, humidity $H = \{h_0, h_1, h_2, h_3, \dots, h_{10}\}$, wind speed and direction $W = \{x, y, z\}$ and fire $F = \{f_{yes}, f_{no}\}$. According to these parameters six network states are defined (see **Table 6-3**).

Table 6-3: Possible states of our collaborative group-based WSN.

State	Definition
S_0	No fire + No wind + High humidity (≥ h_5) + Low temperature
S_1	No fire + No wind + Low humidity (≤ h_5) + High temperature
S_2	No fire + Wind + Low/high humidity + High temperature
S_3	Fire + No wind + High humidity (≥ h_5) + Low temperature
S_4	Fire + No wind + Low humidity (≤ h_5) + High temperature
S_5	Fire + Wind + Low/high humidity + High temperature

So, when a sensor node measures one of these states, it performs the required action for each state (see **Table 6-4**). These actions are the most suitable for the collaborative group-based WSN considered. It implies improving the performance of these networks and energy saving because three levels of alert are considered.

So, each state is performing the most appropriate action for the network. In order to do this process, we used a decision system which is based on the uncertainty. The decision maker knows the possible states, but it does not have any information about which of them is the best state to be changed. Not only it is unable to predict next state, but also it cannot quantify in any way this uncertainty. In particular, this excludes knowledge of probabilistic information on the possibilities of occurrence of each state. In order to develop the decision criterion it has to know the matrix of criteria (see **Figure 6-26**), where in each box is defined by probability of performing an action for a state. It will be performed by each node, e.g. brown node in **Figure 6-27**.

Table 6-4: Actions required.

Action	Network State	Definition
A_0	S_0	No alert
A_1	S_1	Sensor node sends the alert level 1 to nodes of the same group
A_2	S_2	Sensor node sends the alert level 1 to nodes of the same group and to neighboring groups in the same direction of the wind
A_3	S_3	Sensor node sends the alert level 2 to nodes of the same group
A_4	S_4	Sensor node sends the alert level 3 to nodes of the same group
A_5	S_5	Sensor node sends the alert level 3 to nodes of the same group and to all neighboring groups

		States					
		S0	*S1*	*S2*	*S3*	*S4*	*S5*
Actions	*A0*	$x_{0,0}$	$x_{0,1}$	$x_{0,2}$	$x_{0,3}$	$x_{0,4}$	$x_{0,5}$
	A1	$x_{1,0}$	$x_{1,1}$	$x_{,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$
	A2	$x_{2,0}$	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$	$x_{2,5}$
	A3	$x_{3,0}$	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$	$x_{3,5}$
	A4	$x_{4,0}$	$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$	$x_{4,5}$
	A5	$x_{5,0}$	$x_{5,1}$	$x_{5,2}$	$x_{5,3}$	$x_{5,4}$	$x_{5,5}$

Figure 6-26: Matrix of criteria.

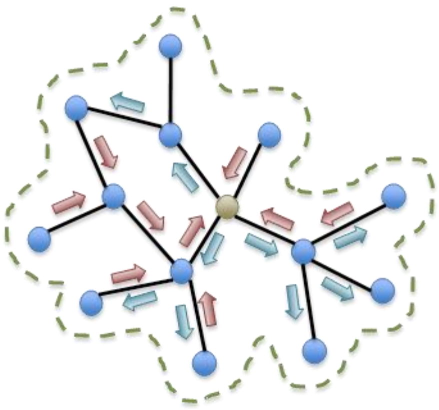


Figure 6-27: A group of our proposed WSN.

The matrix of criteria is based on the criterion of Savage [8], which indicates that the decision maker compares the result of an action under a state with all other outcomes, regardless of the state under which they occur. However, the state is not controllable by the decision maker, so that the result of an action should only be compared with the results of the other alternatives under the same state of nature. For this purpose, Savage defines relative loss or loss of opportunity $r_{i,j}$ (see Eq. 6-12) associated with a result $x_{i,j}$ as the difference between the result of the best alternative because *Si* is the true state and outcome of the action A_i under the state S_i.

$$r_{i,j} = \max_{1 \le k \le m} \{x_{k,j}\} - x_{i,j} \qquad \text{Eq. 6-12}$$

But the Savage criterion propose to select the action that provides the smallest of the major losses suffered (ρ_i), i.e., if r_i is defined as the greatest loss that can be obtained by selecting the action A_i (see Eq. 6-13).

$$\rho_i = \max_{1 \le j \le n} \{r_{i,j}\} \qquad \text{Eq. 6-13}$$

For the application of this criterion, the node calculates the matrix of relative losses which consists of the $r_{i,j}$ elements. Each column of this matrix is obtained by calculating the difference between the maximum value of that column and each one of the values listed. For selecting the best action in each state we use the Eq. 6-14.

$$r(A_i, S_j) = \begin{Bmatrix} max_{a_k}\{x(A_k, S_j) - x(A_i, S_j)\} & Good\ decision \\ x(A_i, S_j) - min\{x(A_k, S_j)\} & Bad\ decision \end{Bmatrix} \qquad \text{Eq. 6-14}$$

6.6.1 Simulation of the Collaborative Decision System

In order to evaluate the accuracy of our collaborative decision system we simulated it. We took the example provided in **Figure 6-27**. This simulation has been done using Matlab [104]. In each case when an event happens in a sensor node, it will send an information request to its neighbors, which will reply with their information and, after taking a decision based on the information received, it will reply with the decided action. **Figure 6-28** shows protocol procedure. We have defined 20 possible cases making a matrix of criteria. Then, we applied our decision system based on the Savage criterion during 100 times. **Figure 6-29** shows the estimated average for each case. In this figure we can see the action selected by each state. The best solution in each state is the action with the same subscript, i.e., $S_n \rightarrow A_n$. We can see in **Figure 6-29** that our system is not perfect, but it only has an error of 3.52%.

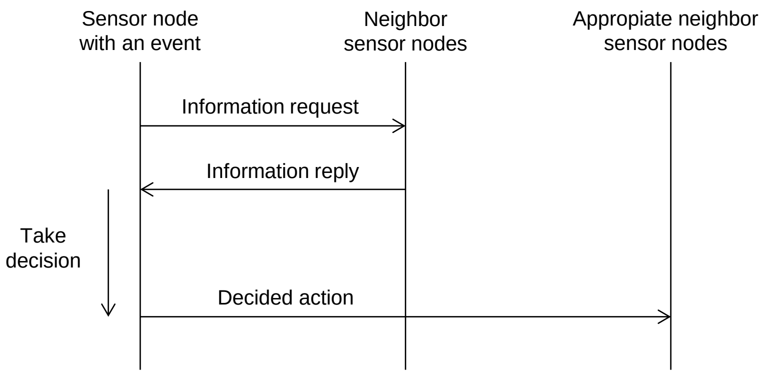


Figure 6-28: Protocol procedure.

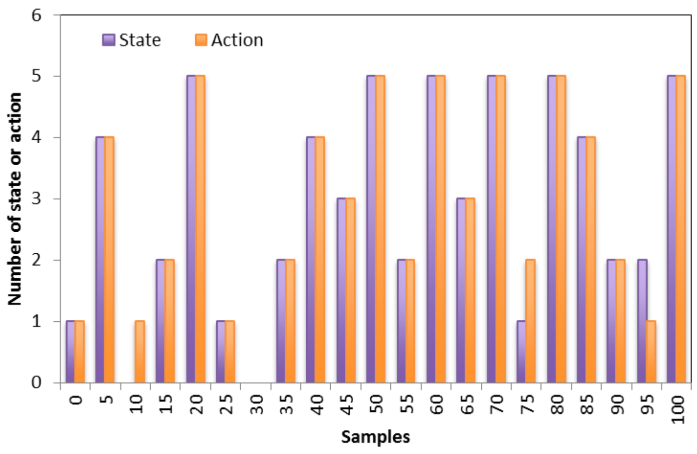


Figure 6-29: Collaborative decision system simulation.

6.7 Conclusion

In this chapter, we have shown three different methods for improving the performance of WSN through saving energy.

First of all, we have explained the different architectures developed to support WSNs. Up to now, regular WSNs have been the most used in a real environment, because they are deployed easily. But, they are not the most efficient from the point of view of the lifespan of WSNs because they are not developing taking into account the energy consumption. However, we consider essential to incorporate mechanisms in the functioning of these networks to reduce this consumption and enlarge their lifespan. Energy saving is absolutely necessary in some applications, for example if a WSN is deployed in an environment with difficult or danger access, it is necessary that it works for a long time or as much as possible.

So, in this chapter we have presented our proposal to achieve it: collaborative group-based WSNs. We introduce the main idea of our proposal. It is based on allowing different levels of alert depending on the degree of danger of the sensed parameter and the position of the neighboring groups. In this type of WSNs when a sensor detects a new event, the alert is sent to its group and it is distributed to an appropriate neighboring groups based on the information shared between sensors. Cooperation between groups could be used to change the direction of the alert propagation and the level of the alert in order to take the appropriate actions.

Using several analytical analyses we have proved that the cooperative group-based WSNs save energy and improve the efficiency of the WSN communications. Moreover, we have seen that there are some WSN topologies that have better results than others.

Then, a new intragroup communication has been presented. This is based on a collaborative intergroup routing pool tree where the end nodes exchange information about the neighbor's groups. This exchanged data creates a collaborative routing pool tree where the root is the sink node and there is collaboration between the neighbor's groups. This intragroup communication improves the energy efficiency as we have seen on the previous simulations.

At the end of this chapter we propose a cooperative decision system based on several parameters for selecting the better action in a group-based WSN for environmental monitoring. According to this decision, the sensor node sends the appropriate level alert to its group or to the appropriate neighboring groups. Simulation shows that the accuracy of the criterion is quite good because the system only has an error of 3.52%.

Finally, work presented in this chapter has been published in the following references [21], [22], [23], [24] and [25].

Chapter 7

SECURITY AND INTRAGROUP TIME SYNCHRONIZATION IN GROUP-BASED WSNS

7.1 Introduction

WSNs are used in several application fields [32]. For instance, in energy efficiency control, high-security environments, environmental monitoring, industrial control, automotive industry, medical field, etc. The implementation of these networks is following an exponential growth [105] due to their own features. On the one hand, these networks are deployed easily and they are configured by themselves. So, each sensor node can become a transmitter, receiver or gateway between two other nodes without direct sight or register information of neighboring nodes at any time. On the other hand, they support energy efficient management which allows them to get a high rate of autonomy to work by themselves.

In order to deploy a WSN, it is necessary to analyze its lifespan, coverage, budget, the best way to install it, the suitable response time, accuracy and frequency of measurements and security. Moreover, in order to select the most suitable sensor nodes, we should take into account its energy, way of communication, processing capacity, way of synchronization, size and cost [33]. Nowadays, many of WSN's applications must not be implemented without taking care of some security issues. The WSNs cannot use the same security methods than other networks due to their limitations. The WSN limitations that must be taken into account when a security system is wanted to be included are [106]:

- Very low power consumption: The batteries of the nodes must have long lifetime. For this reason, the processing time and the data transmission must be low.
- Very limited local memory and process capacity: The security process should not need high processing capacity or a lot of memory. The algorithms to encrypt and decrypt should be simple and effective.

- Large network: The security architecture should be scalable. Most of the WSNs are formed by many nodes. For this reason, the security should not imply high overhead to the system.
- Fault tolerance: The WSNs tend to be deployed in hard environments where the likelihood of failure is quite high. The security algorithms must be designed to operate even if a node fails.
- Bandwidth and network coverage are limited, so the way of communication, transmission and reception of data is different in such networks.

Therefore, the main feature to consider in these networks is the energy saving. This is the reason why the nodes are awake in short periods of time, only when they are taking data or measuring some variable. It depends on the frequency between measurements.

Synchronization is important for several distributed systems [107]. In particular it is essential for an efficient use of energy, data fusion, localization and many other applications. Synchronization algorithms for traditional networks are inefficient for WSNs, for instance features assumed by the traditional time protocol NTP (Network Time Protocol) [72]. They are not applicable for WSNs. It assumes a static network topology where nodes are installed before the synchronization, configured by hand and the delay between nodes can be estimated with high precision. In contrast, WSNs have dynamic topologies and nodes can be attached to anything: animals, buoys, plants, etc., and anytime new nodes can be added to the network. Features of WSNs cannot be predicted in advance and energy is a resource very limited. So, although synchronization of nodes is essential, it is difficult to get it. Firstly, we did an initial study [108] about the synchronization of multimedia groups in order to know the current schemes of synchronization.

The synchronization protocol should use low energy, be scalable, robust and assume an ad-hoc dynamic topology. Moreover, sensor network is so vulnerable to attacks that both an internal and external adversarial can easily corrupt the synchronization mechanism in sensor networks.

Taking into account their features, in this chapter we are going to propose a new secure system for group-based WSN communications, which will help to have a secure communication and synchronization system. Firstly, we are going to show a section with some works related to security and time synchronization in WSNs. Then, we will describe the secure algorithm that is proposed for a group-based WSN. Moreover, its operation and hey distribution system will be explained and finally we will check the energy consumed when this secure system is running. The following issue will be to know the techniques and possible errors of time synchronization in WSNs. Afterward we will explain our algorithm based on principles of receiver-receiver synchronization protocols, but it includes security (every message sent in this algorithm is encrypted according to our security proposal). In the next section we will show a mathematical study to know the offset times between sensor nodes. Several simulations are presented in order to test our synchronization algorithm. Finally, we present our conclusions about the solutions presented in this chapter.

7.2 Group-based Secure Algorithm

7.2.1 Secure Network

Our proposal has two security zones: the intragroup security and the intergroup security. The intergroup security is related with the secure communications between different groups of the same network. In contrast, the intragroup security keeps the security inside the group (see **Figure 7-1**). The intergroup security is based on the symmetric cryptography with a single key for all members of the network. This feature has been chosen due to the limited local memory and processing capacity of sensor nodes in WSNs. The data are sent inside the group using a shared key when the information should be known by all members of the group.

In contrast, the intragroup security is different. In this case, a node can belong to a group, and, later, it could change to another group because network topology has changed. For this reason, the keys have to be different for each group. Communications within each group are closed and only known by the nodes that belong to that group. When the information is sent to another group, it is encrypted using a shared key that is known by all the nodes of the network.

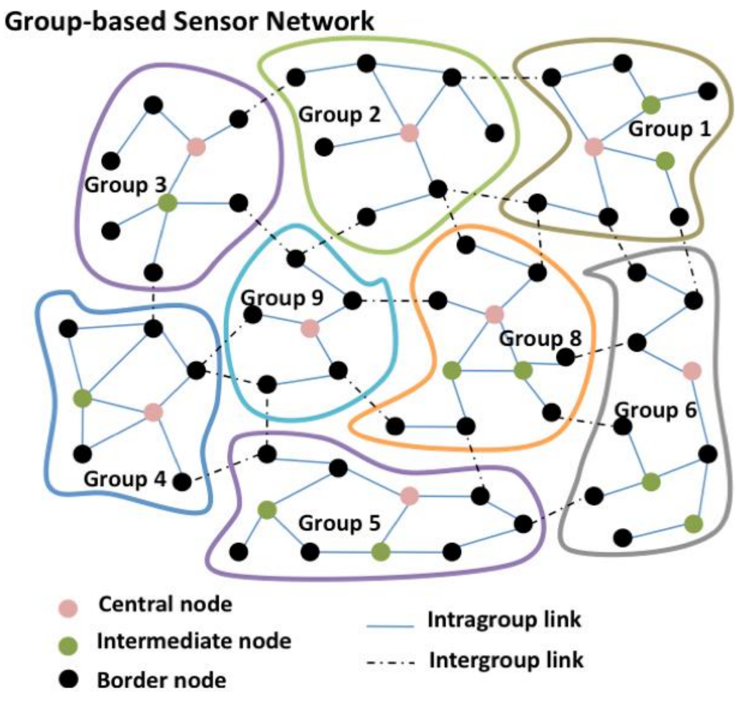


Figure 7-1: Example of a group-based WSN topology.

The communication is more effective because the nodes use less cycles to process the security and therefore they consume less energy. When a new sensor is going to join the network, first, it has to be authenticated from any sensor of the network. Every sensor node of our network has a triple (*Kn, Kg(i,j), P*). The meaning of each parameter is shown in **Table 7-1**.

Table 7-1: Meaning of the parameters of the security triple.

Parameters	Definition
Kn	Network key
Kg(i,j)	Node (i,j) group key
P	Relationship of update among the group keys

7.2.2 Encryption Algorithms

For the intergroup encryption we have used symmetric cryptography. Each node of the network has the network key (*Kn*). They acquire *Kn*, when they join the network, from the node that authenticates them. This key is stored in the memory of the sensor node until it leaves the network. Each node of the network knows the *Kn*.

We have chosen a simple symmetric encryption for intergroup communication. When a node has a message M that wants to send it to other node, the message is encrypted using the *Kn* key. The procedure is shown in **Figure 7-2**. The operation result is denoted as the encrypted message *M'*. At the same time, the origin node makes the MD5 hash of message *M*, obtaining m. *M'* message plus the hash function (*m*) are sent to the receiver node. Then, the receiver node decrypts *M'* with the key *Kn* and it checks the integrity of *M'* comparing the received *m* with the calculated *m*.

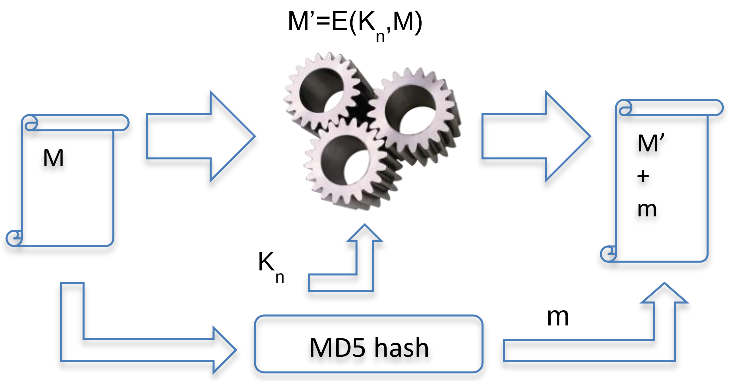


Figure 7-2: Symmetric encryption process.

The intragroup communication security is more complex than the intergroup one. There are different parameters that must be taken into account. They are:

- Group size
- Characteristics of the members
- Group dynamics
- Group lifetime
- Number and type of issuers
- Volume and type of traffic

In addition, a member should not be able to access the network information before it joins the group; neither the nodes should not be able to access the information after they leave the group.

The head node of each group is responsible of managing the group key. It is also responsible of saving the key logical tree structure of the group members. An example is shown in **Figure 7-3**.

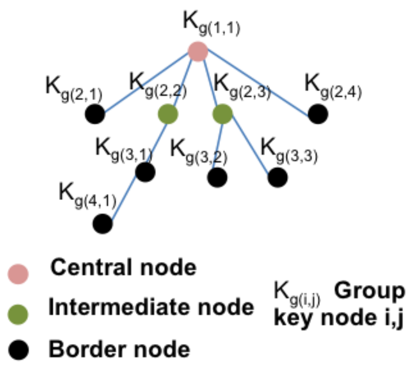


Figure 7-3: The key logical tree structure of a group.

The key logical tree structure is made using our algorithm proposed in Chapter 4. The key tree is created when the group is being formed. Each end node knows the keys from it to the head node. The operation is shown in the next subsection.

In order to reduce the memory wasted by the head node (this node must store and maintain the key tree), we use pseudorandom functions to generate the keys. For that reason, a node only needs to know some rules, but not all keys. The group key of a node (*i,j*) is defined by Eq. 7-1.

$$K_{g(i,j)} = F_p \cdot \ (2^i + j) \bigoplus \mathrm{r} \qquad \text{Eq. 7-1}$$

Where the group key of the node (*i,j*) is equal to a pseudorandom function (*Fp*) plus an update factor *r*. A pseudorandom function is a family of functions with the property that the input-output behavior of a random instance of the family is "computationally indistinguishable" from that of a random function. The operation mode is as follows. Let us suppose that the key *Kg(i,j)* must be updated. The head node must send P to all nodes that had stored the key *Kg(i,j)*. *P* function is given by Eq. 7-2.

$$P = r \oplus \mathrm{r}' \quad \text{Eq. 7-2}$$

Each authorized node will calculate the new key through Eq. 7-3.

$$K_{g(i,j)} = K_{g(i,j)} \oplus \mathrm{P} = F_p \cdot \ (2^i + j) \oplus \mathrm{r} \oplus r \oplus \mathrm{r}^{'} = F_p \cdot \ (2^i + j) \oplus \mathrm{r}' \quad \text{Eq. 7-3}$$

When all upgrades have been finished, all keys have the structure shown in Eq. 7-4. Using this method, the head node only must save in memory the used pseudorandom function (*Fp*) and the update factor (*r'*).

$$K_{g(i,j)} = F_p \cdot \ (2^i + j) \oplus \mathrm{r}' \quad \text{Eq. 7-4}$$

The proposed secure architecture has some security extra considerations. On one hand, in order to avoid conspiracy of the revoked members, Eq. 7-5 must be fulfilled. On the other hand, in order to avoid conspiracies of the authorized members, Eq. 7-6 should be fulfilled.

$$F_p(x) \oplus r_i^{'} \oplus F_p(y) \oplus r_i^{'} \neq F_p(n) \quad \text{Eq. 7-5}$$

$$x, y \ \epsilon \ R; n \ \epsilon \ N - R$$

$$r_i' \oplus r_j' F_p(y) \oplus r_i' \neq F_p(n) \quad \text{Eq. 7-6}$$

$$x, y \ \epsilon \ R; n \ \epsilon \ N - R$$

The encryption method used in all our network is based on MISTY1 [109]. MISTY1 is specifically designed to resist differential and linear cryptanalysis. It is designed for high-speed implementations on hardware as well as software platforms by using only logical operations and table lookups.

We find MISTY1 to be particularly suitable for 16-bit platforms. It is a royalty-free open standard documented in RFC2994 [110]. We have chosen this encryption method because we took into account the study performed by Yee Wei Law et al. in [111]. In this work, several encryption methods were analyzed using encryption blocks in WSNs. In conclusion, the authors indicated that MISTY1 is the most efficient encryption bearing in mind energy terms. It is a recommendable option if we have nodes with little memory, but we need high security.

7.3 Operation and Key Distribution

In this subsection we explain the formation of the secure architecture. In the first part, we will see how the network and the groups create a secure architecture and, in the second part, we will see the tasks carried out when a node leaves a group (or it fails).

7.3.1 Secure Group Creation and its Maintenance

From the outset, each node knows its position (x,y) obtained by using GPS or a wireless location system [13] and the basic pseudorandom function Fp. When a new node joins the network (it could be the first), it sends a hello message with its position (x,y) (called helloGroup) in order to join a group. If there is no response from any node for 3 seconds, the node considers itself as a head node of a group, and it will take the value groupID=n and nodeID=x1,y1. Each head node can calculate its function *Fp*, this function depends on the *groupID*, therefore it will be different for each group.

If a new node receives a reply, given by the message *helloGroupACK*, it will receive the update factor *r* and the group identifier *groupID=n*. There are two possible situations, the first one is when new node is directly connected to a head node, and the second one is when there are leaf nodes. The process to obtain a secure communication in the first situation is seen in **Figure 7-4**. New node sends the *helloGroup(x,y)* message with the information of its position.

The head node saves that position and sends the *helloGroupACK(r,n)* message, where r is the update factor and n is the index to use in the modified function (*Fp*) for that group. While the head node sends the *helloGroupACK* message, it estimates the key for the new node (see Eq. 7-1). The new node performs the same action once it receives the *helloGroupACK* message. When the new node has the key, it encrypts the *okGroup* message with the key *Kg(x,y)*, then this message is sent to the head node. The head node decrypts the message and replies to the new node with the message *okGroupACK(Kg(i,j))* encrypted with *Kg(x,y)*. The new node belongs the group n, it holds its key, *Kg(x,y)*, and all the keys to its branch from the new node to the head node, within the key logical tree.

In second situation, when there are leaf nodes, the process is very similar to the previous one. Due to the existence of leaf nodes, their intermediate keys must be updated. Thus, the process to obtain the group key will have more delay than the process seen in **Figure 7-4**. The process can be seen in **Figure 7-5**. When a new node joins into a group (*groupID=n*), this node sends the message *helloGroup(x,y)* to its neighbor node, this neighbor node routes this message to the head node. When the head node receives this information, it sends a *helloGroupACK(r',n)* with *n* (group identifier) and *r'* (the new update factor for all nodes of this branch). At the same time, the head node begins to calculate the *Kg(x,y)* of the new node. When the new node receives the *helloGroupACK(r',n)* message, it encrypts and sends the message *okGroup* to the head node. This node decrypts the message to verify that the process has been done correctly.

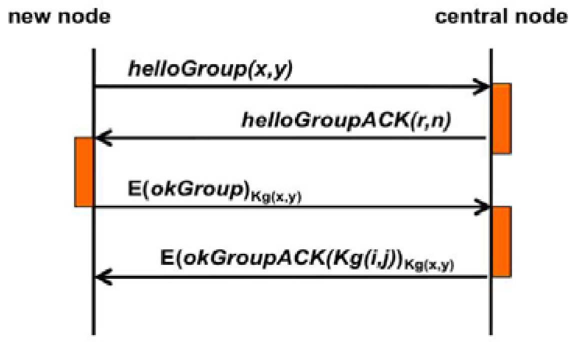


Figure 7-4: Key Exchange between a new node and the head node.

Head node sends encrypted messages *okGroup(Kg(i,j), Kg(k,l), Kg(m,n))* with the key to new node. Thus, the new node will keep all the keys of its branch in the key logical tree. Finally, the new node will send an *okGroup* message to the selected neighbors, and the neighbors will reply with the *okGroupACK* message with the keys of their branch indicating that the link has been established. Nodes will send *keepalive* messages periodically to their neighbors. If a node does not receive a *keepalive* message from a neighbor before the dead time, it will remove this entry from its database and will start the group update process. The update process is very similar to the one described before, but in this case the messages will distribute P (see Eq. 7-2).

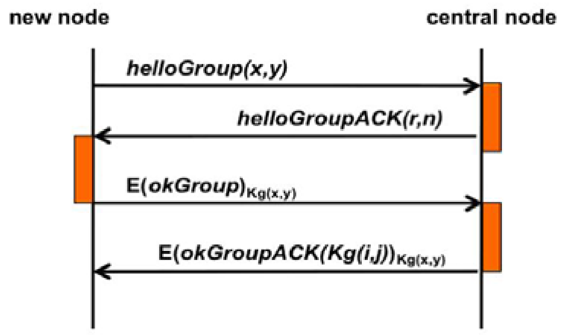


Figure 7-5: Key exchange for new node when there are leaf nodes.

7.3.2 Key Management when a Node Changes to Another Group

When a node leaves the group, it sends *nodeDisconnect* message to its neighbor nodes. Then, they must reply with a *nodeDisconnectACK* message and send the *nodeDisconnect* message to the head node. The process is shown in **Figure 7-6**. The head node distributes update information using RPF algorithm. If a neighbor node does not have links with other neighbors, it must start a new connection process.

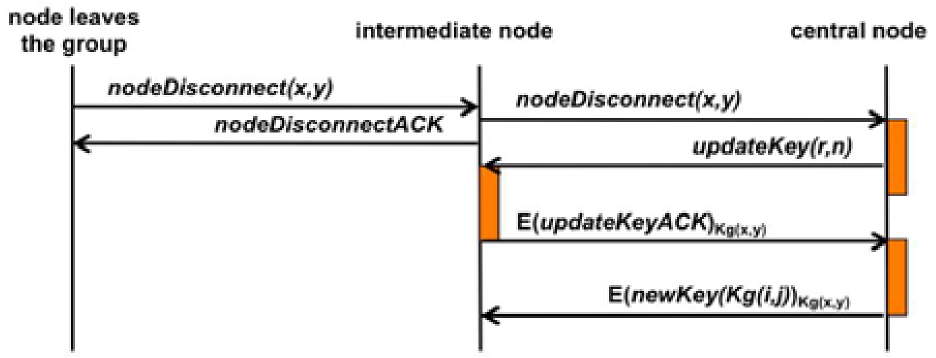


Figure 7-6: Keys update when a node leaves its group.

If the leaving node is the head node, it assigns the head node role to the most appropriate candidate. This decision is taken using the value of the diameter of the group (the head node knows how many hops are to the farthest nodes in its group). In case of draw, it will choose the oldest one in the group. Then, it sends a changeHead message to the nodes of the group in order to inform them about its action, and leaves the group. All the messages sent are shown in **Figure 7-7**. When a node fails down, its neighbor nodes will know the failure because of the absence of its keepalive messages. The procedure performed in this case is the same as when the node leaves the network voluntarily.

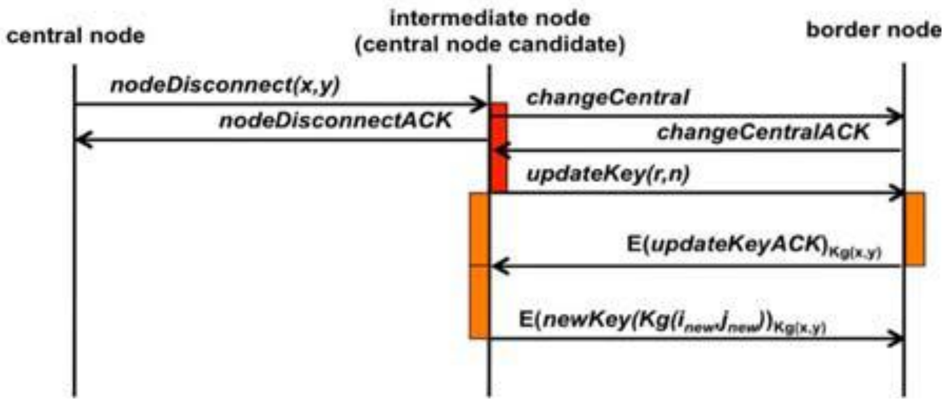

central node
intermediate node (central node candidate)
border node
nodeDisconnect(x,y)
nodeDisconnectACK
changeCentral
changeCentralACK
updateKey(r,n)
E(updateKeyACK)$_{Kg(x,y)}$
E(newKey(Kg(i$_{new}$,j$_{new}$))$_{Kg(x,y)}$

Figure 7-7: Key management when a head node leaves its group.

7.4 Measurements

In this section we estimate the energy consumption given by the cases explained in the previous section. Number of CPU cycles using MISTY1 encryption algorithm has been taken from [111]. We have selected operation counter mode (CTR) which is very similar to output feedback mode (OFB). The values obtained in this section are tentative values, because these values have been extracted from the EYES sensor node [112] and it could be different if we use any other sensor. The packets sent have an average size of 128 bits. A sensor node uses 3 million instructions to transmit a packet of 1 KByte to a distance of 100 meters. In our case, we have 128 bits, so there will be 0.375 million instructions. A sensor node consumes 1 watt every 100 million instructions, therefore in this case the sensor node will consume 3.75 mW, that is, 3.75 mJ/s. On the other hand, every time a packet arrives, 615 CPU cycles are used to decrypt or encrypt a message. Each CPU cycle consumes 1.26 nJ, therefore 0.775 µJ/s will be consumed to encrypt a packet each time.

In **Figure 7-8**, we can see the energy consumed by each situation described in the previous section. We can observe that the situation which consumes most energy is when a new node joins a group and the neighboring node is not a neighbor of the head node. It consumes 33.7531 mJ. The best case is when there is only one node between the new node and the head node. This energy consumption decreases more than a half if the new node is a neighbor of the head node (approximately 15 mJ).

Energy cost when a node, which is not the head node, leaves the group is approximately 18.75 mJ. If the node that leaves the group is the head node, the energy consumption is 26.25 mJ in the best case. In this situation, several messages are transmitted. In addition, the nearest node to the head node must estimate who will be next head node of the group (see Figure 8). Finally, our system has been compared with other secure systems used in wireless sensor networks. The measurements obtained in [21] for the several secure mechanisms are calculated for two sensor nodes (MICAz and TelosB). These sensor nodes are different from our sensor node (EYES). For this reason, using the measurements provided in [112], we have estimated the average values in order to avoid the heavy reliance of the measurements.

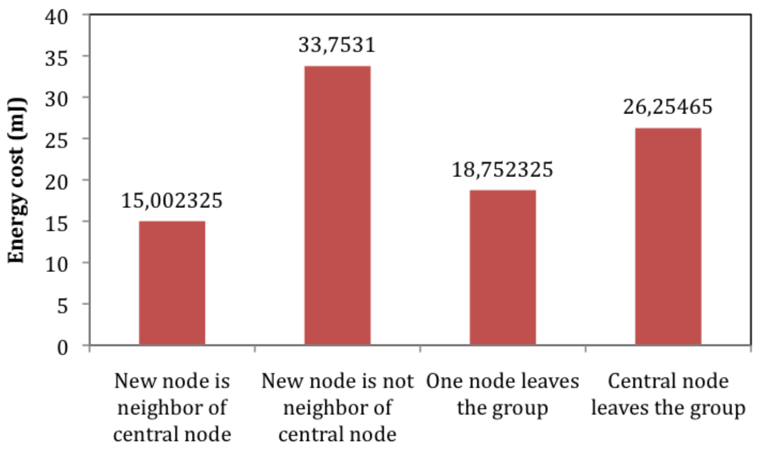


Figure 7-8: Energy cost in each process of the previous section.

In **Table 7-2** we can see that our proposal has obtained a good score in the column “Energy cost average”. It is the third value with only 0.25 mJ more than the system “ECDSA-160 sign”. The “AES-128” has the best score because this system uses symmetric encryption and it does not perform key exchange. Our system is used for group-based wireless sensor networks and as we can see in **Table 7-2**, the energy cost for a good performance is lower or equal than the cost achieved by the other systems.

Table 7-2: Energy cost comparison.

Secure algorithm	MICAz	TelosB	EYES	Energy cost
AES-128	**38 µJ**	**9 µJ**		**23.5 µJ**
ECC-160 point mult	**55 mJ**	**17 mJ**		**36 mJ**
ECDSA-160 sign	**52 mJ**	**15 mJ**		**33.5 mJ**
ECDSA-160 verify	**63 mJ**	**19 mJ**		**41 mJ**
Our proposal (worst situation)			**33,75 mJ**	**33,75 mJ**

7.5 Time Synchronization: Techniques and Features

WSNs have the need to coordinate the communication, computing, sensing, and performance of distributed nodes, so an accurate and consistent system to synchronize them is essential to keep the sensor network in working order. This section discusses about the main motivations for time synchronization, the challenges that it involves and some solutions to solve them.

7.5.1 Why a WSN needs Time Synchronization?

WSNs need time synchronization among sensor nodes due to several reasons.

Here is a list with the main ones:

1. Time-stamping in measurements. Even the simplest data collection often requires readings with date/time and location information. This is especially important when a significant delay between the transmitter and the receiver node or the base station is possible.
2. Signal processing in the network. Date/time information is necessary to determine which information from several sources can be added within the network. Many collaboration algorithms of processing signals, like tracking phenomena or unknown objectives, require a constant and exact synchronization.
3. Localization. There are several techniques in order to locate nodes, which use the propagation time required to transmit a message to a reference node, so these techniques need proper time synchronization.
4. Cooperative communication. Some techniques of multi-node cooperative communication involve multiple transmitters broadcasting in phase to a given receiver. Such techniques have the potential to provide energy savings and significant strength, but require precise timing.
5. Media Access. Schemes based on TDMA (Time Division Multiple Access) require node synchronization to assign them to different slots and guarantee collision-free communication.
6. Sleep scheduling. One of the most significant sources of energy savings is the inactivity of the radio device, which is called sleep state. However, the synchronization is necessary to coordinate the sleep schedules of neighboring devices, so they can communicate with each other efficiently.
7. Coordinated action. In advanced applications in which the sensor network includes distributed sensor nodes, besides detection it is necessary the synchronization to coordinate the distributed sensor nodes with distributed control algorithms.

7.5.2 Synchronization Types used in WSNs

There are different ways to classify the synchronization techniques in WSN, but it is often represented by several pairs of contrasting distinctive features

[113]. Synchronization algorithms can be proactive or reactive. The proactive algorithms carry out synchronization tasks continuously to keep the offset delay bounded. For example a reference node sends periodically a message with its time-value to other nodes. This allows knowing the timing offset respect to the reference time. Reactive algorithms act when an event is detected; a node sends its time to the base station that performs the necessary corrections.

Moreover, these synchronization schemes are divided into adaptive and non-adaptative. Adaptive schemes can modify important parameters of the synchronization procedure, such as frequency, response to certain changes in the network or the environment. On the other hand, non-adaptive schemes never change their way of working. Synchronization schemes introduce computational overhead. They increase network traffic due to exchange messages and this means more power consumption. It is important to maintain a balance between the disadvantages to incorporate a synchronization system in WSNs and the advantages of keeping bounded the errors introduced by offset time clocks in the sensor nodes.

Among the most common synchronization procedures we can mention:

- Sender-receiver synchronization: Pairwise sender-receiver synchronization is performed by a handshake protocol between a pair of nodes. This protocol is executed in three steps (see **Figure 7-9 a**). *T1*, *T4* represent the time measured by the local clock of node A. Similarly *T2*, *T3* represent the time measured by *B's* local clock. At time T1, *A* sends a synchronization pulse packet to B. Node *B* receives this packet at *T2*, where *T2* is equal to *T1+δ+d*. Here, *δ* and *d* represent the offset between the two nodes and end-to-end delay respectively. At time *T3*, *B* sends back an acknowledgement packet. This packet contains *T2* and *T3*. Node *A* receives the packet at *T4*. Similarly, *T4* is related to *T3* as *T4=T3-δ+d*. Node A can now calculate the clock offset and the end-to end delay.
- Receiver-receiver synchronization: In receiver-receiver model, it synchronizes receivers by recording the arrival time of a reference message in a broadcast medium, and then exchanging this arrival time to determine the discrepancy between the respective clocks of the receivers. This technique is shown in **Figure 7-9 b**. The approach avoids various sources of synchronization error because a broadcast message is received at exactly the same time by all receivers, at least in terms of the granularity of their clocks.
- WSN nodes can be synchronized either to some external time reference, or by means of some intra-network synchronization procedures. A GPS receiver can serve as an external source of time for a wireless network, where the time values can be disseminated by a special node attached to the GPS receiver. In this case, network nodes are able to transform their time values to the reference time values and vice versa, thus achieving the time synchronization. Alternatively the network can be synchronized internally, i.e. using the clock of some node as a reference clock.

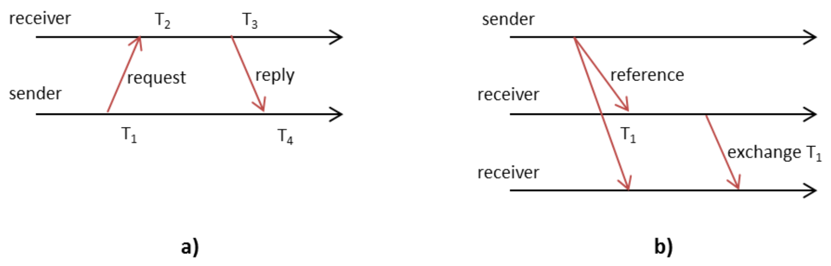


Figure 7-9: a) Sender-receiver synchronization. b) Receiver-receiver synchronization.

7.5.3 Error Sources in Time Synchronization

Time synchronization algorithms in WSNs use the measured time in the exchange of messages between nodes to synchronize the clocks on different nodes, estimating their offsets. An important aspect that we must take into account when we use algorithms based on the exchange of messages is that this exchange generates delays. Moreover of these delays, the exchange of messages can be corrupted, so they could inject errors in the synchronization algorithms. In order to know the origin of these errors we should identify some phenomena that take into the process for sending data. The most important are:

- Send Time: It is the time used to load the message and to make the request and to send it to the MAC layer in the transmitter side. Depending on the overhead of system and the current processor load, the send time is not deterministic and it can be as high as tens of milliseconds.
- Access Time: It is the delay caused by the access to transmission channel at the beginning of transmission. Access time is the least deterministic message when a message is delivered in a WSN. It varies from tens to hundreds of milliseconds, depending on the current network traffic.
- Transmission Time: It is the time used by the transmitter to send a message. It is around ten milliseconds, but it depends on the length of the message and the transmission speed.
- Propagation Time: It is the time required by the message to propagate it from sender to receiver. This time is quite deterministic in WSN and it only depends on distance between nodes.
- Reception Time: It is the time taken by the receiver to receive the message. It equals than transmission time. The transmission and reception times can overlap in WSN.
- Receive Time: It is the time required to process the incoming message and notifying to this receptor. Its characteristics are similar to Send Time.

- Attackers can be either outsiders or insiders. The outsiders do not possess keys for encryption and authentication. Thus, they can only interfere in the transmission of time synchronization messages, but not modify or generate those messages.

On the other hand, the compromised sensors, which are insiders, can manipulate the time synchronization protocols in many ways.

7.6 Intragroup Secure Time Synchronization

Our proposal is based on RBS protocol [76]. Our objective is that our algorithm exchanges fewer timing messages between sensor nodes, which means saving energy. RBS protocol requires a super node, which broadcasts a time reference message, so it is a centralized protocol. In contrast, in our proposal all nodes send messages about timing information. So, there is not a super node which manages the synchronization. We propose a receiver-receiver synchronization scheme, where each message exchanged between nodes inside a group is encrypted. The encryption keys are explained in the previous section. Each sensor node encrypts its message and any node of logic branch that joins with head node can decode this message. Moreover, all nodes can decode messages encrypted by head node. For example, sensor node in Fig. with the key (*Kg(4,1)*) encrypts a message. This message can be decoded by all nodes connected to the head node. In contrast, when this message arrives to this one and it should be sent to other branch, the message should be encrypted.

In order to explain our algorithm we use an example scheme illustrated in **Figure 7-10**. It shows our intragroup receiver-receiver synchronization scheme. We suppose that the group identification is *group_ID=1*. Firstly, the head node (HN_1) starts the k^{th} synchronization process sending a synchronization beacon ($Beacon_{syn}$) with the information $T^1_{syn,k}$.

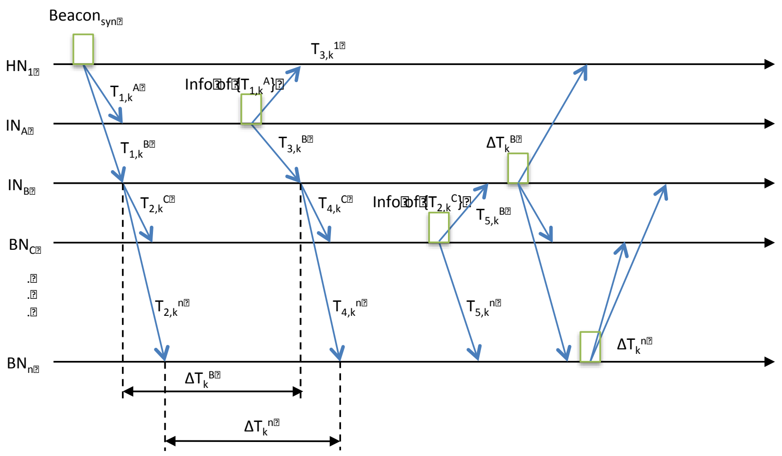


Figure 7-10: Intragroup synchronization scheme.

This message only arrives to sensor nodes under head node coverage. These nodes record the message arrival time, following our example $(T_{1,k}^{A}, T_{1,k}^{B})$. These times are explained on Eq. 7-7 and Eq. 7-8, where $T_{syn,k}^{1}$ is a reference time, φ_{offset}^{1-A} is clock offset between the node HN$_1$ and the node IN$_A$, δ_{prop}^{1-A} is the propagation delay between both nodes and τ_{rx}^{1-A} is the receive delay at these nodes. Then, these nodes (IN$_A$, IN$_B$) resend this message to others nodes (BN$_C$, BN$_n$), which are farther from the head node. These last nodes record the new times $(T_{2,k}^{c}, T_{2,k}^{n})$, which are detailed on Eq. 7-9 and Eq. 7-10. In these cases, both equations depend on $T_{1,k}^{A}, T_{1,k}^{B})$.

$$T_{1,k}^{A} = T_{syn,k}^{1} + \varphi_{offset}^{1-A} + \delta_{prop}^{1-A} + \tau_{rx}^{1-A} \qquad \text{Eq. 7-7}$$

$$T_{1,k}^{B} = T_{syn,k}^{1} + \varphi_{offset}^{1-B} + \delta_{prop}^{1-B} + \tau_{rx}^{1-B} \qquad \text{Eq. 7-8}$$

$$T_{2,k}^{C} = T_{1,k}^{B} + \varphi_{offset}^{B-C} + \delta_{prop}^{B-C} + \tau_{rx}^{B-C} \qquad \text{Eq. 7-9}$$

$$T_{2,k}^{n} = T_{1,k}^{B} + \varphi_{offset}^{B-n} + \delta_{prop}^{B-n} + \tau_{rx}^{B-n} \qquad \text{Eq. 7-10}$$

Then, node IN$_A$ sends a message similar to Beacon$_{syn}$ with the information $T_{1,k}^{A}$to HN$_1$ and IN$_B$. They record the arrival time in each node $(T_{3,k}^{1}, T_{3,k}^{B})$. IN$_B$ resends this information to BN$_C$ and BN$_n$. These last nodes record also the times $(T_{4,k}^{c}, T_{4,k}^{n})$. Similarly, BN$_C$ sends a beacon with information of $T_{2,k}^{c}$. This beacon is received by IN$_B$ and BN$_n$, so they record the times $T_{5,k}^{B}, T_{5,k}^{n}$ too. Finally, nodes IN$_B$ and BNn, selected randomly in this example, calculate the differences ΔT_{k}^{B} and ΔT_{k}^{n}. With these data we can know the offset times between these nodes, as we show as follows.

We assume that one process can be finished in a relatively short time, so we can neglect the effect of clock skews in one process.

Eq. 7-11 is deduced from Eq. 7-7 and Eq. 7-8. φ_{offset}^{A-B} is the clock offset between node IN$_A$ and IN$_B$, it is used to correct the clock of node B.

$$\varphi_{offset}^{A-B} = \varphi_{offset}^{1-B} - \varphi_{offset}^{1-A} = T_{1,k}^{B} - T_{1,k}^{A} - \left[\left(\delta_{prop}^{1-B} - \delta_{prop}^{1-A}\right) - \left(\tau_{rx}^{1-B} - \tau_{rx}^{1-A}\right)\right] \qquad \text{Eq. 7-11}$$

All sensor nodes are similar, so we can deduce that τ_{rx}^{1-B} is equal to τ_{rx}^{1-A}. So we obtain the Eq. 7-12, where φ_{offset}^{A-B} is the difference between both times plus an increment of propagation delay. This propagation delay can vary from few picoseconds until 333 picoseconds (see Eq. 7-13), it depends on radio coverage of each sensor node.

$$\varphi_{offset}^{A-B} = T_{1,k}^{B} - T_{1,k}^{A} \pm \Delta\delta_{prop} \qquad \text{Eq. 7-12}$$

$$\Delta\delta_{prop} = [0ps, 333ps[\qquad \text{Eq. 7-13}$$

These propagation delays do not contribute significantly to the overall error. According to reference [72], we can model these delay using a Gaussian distributed random function with mean zero and variance σ^2. For these reasons, it could be dismissed. Finally, φ_{offset}^{A-B} can be estimated from this synchronization process from Eq. 7-14.

$$\hat{\varphi}_{offset}^{A-B} = \frac{1}{M}\sum_{k=1}^{M}\left(T_{1,k}^{B} - T_{1,k}^{A}\right) \qquad \text{Eq. 7-14}$$

From equations Eq. 7-9 and Eq. 7-10 we can obtain φ_{offset}^{c-n} (see Eq. 7-15). With the same reasoning explained in Eq. 7-13 and Eq. 7-14, we have a simple unbiased estimator (see Eq. 7-16).

$$\begin{aligned}\varphi_{offset}^{C-n} &= \varphi_{offset}^{B-n} - \varphi_{offset}^{B-C} \\ &= T_{2,k}^{n} - T_{2,k}^{C} - \left[\left(\delta_{prop}^{B-n} - \delta_{prop}^{B-C}\right) + \left(\tau_{rx}^{B-n} - \tau_{rx}^{B-C}\right)\right]\end{aligned} \qquad \text{Eq. 7-15}$$

$$\hat{\varphi}_{offset}^{C-n} = \frac{1}{M}\sum_{k=1}^{M}\left(T_{2,k}^{n} - T_{2,k}^{C}\right) \qquad \text{Eq. 7-16}$$

From the message sent by IN$_A$, we get Eq. 7-17 and Eq. 7-18:

$$T_{3,k}^{1} = T_{syn,k}^{A} + \varphi_{offset}^{A-1} + \delta_{prop}^{A-1} + \tau_{rx}^{A-1} \qquad \text{Eq. 7-17}$$

$$T_{3,k}^{B} = T_{syn,k}^{A} + \varphi_{offset}^{A-B} + \delta_{prop}^{A-B} + \tau_{rx}^{A-B} \qquad \text{Eq. 7-18}$$

Where the used variables have the same meaning as in Eq. 7-7 and Eq. 7-8. Using Eq. 7-17 and Eq. 7-18, we obtain $\varphi_{offset}^{\mathrm{A}-1}$ (see Eq. 7-19).

$$\varphi_{offset}^{A-1} = T_{3,k}^{1} - T_{3,k}^{B} + \varphi_{offset}^{A-B} - \left[\left(\delta_{prop}^{A-1} - \delta_{prop}^{A-B}\right) + \left(\tau_{rx}^{A-1} - \tau_{rx}^{A-B}\right)\right] \qquad \text{Eq. 7-19}$$

Taking into account Eq. 7-11 to know the value of φ_{offset}^{A-B} and Eq. 7-20, we can obtain Eq. 7-21.

$$\Delta T_{k}^{B} = T_{3,k}^{B} - T_{1,k}^{B} \qquad \text{Eq. 7-20}$$

$$\begin{aligned}\varphi_{offset}^{A-1} = T_{\mathrm{A},k}^{1} - \left(T_{1,k}^{B} + \Delta T_{k}^{B}\right) \\ - \left[\left(\delta_{prop}^{A-1} - \delta_{prop}^{A-B} + \delta_{prop}^{1-B} - \delta_{prop}^{1-A}\right)\right. \\ \left. + \left(\tau_{rx}^{A-1} - \tau_{rx}^{A-B} + \tau_{rx}^{1-B} - \tau_{rx}^{1-A}\right)\right]\end{aligned} \qquad \text{Eq. 7-21}$$

Similar to Eq. 7-14 and Eq. 7-16, we can get an estimate of φ_{offset}^{A-1} from Eq. 7-22.

$$\hat{\varphi}_{offset}^{C-n} = \frac{1}{M}\sum_{k=1}^{M}\left(T_{2,k}^{n} - T_{2,k}^{C}\right) \qquad \text{Eq. 7-22}$$

Following the same process explained before we obtain the next equations:

$$T_{4,k}^{C} = T_{3,k}^{B} + \varphi_{offset}^{B-C} + \delta_{prop}^{B-C} + \tau_{rx}^{B-C} \qquad \text{Eq. 7-23}$$

$$T_{4,k}^{n} = T_{3,k}^{B} + \varphi_{offset}^{B-n} + \delta_{prop}^{B-n} + \tau_{rx}^{B-n} \qquad \text{Eq. 7-24}$$

$$T_{5,k}^{B} = T_{syn,k}^{C} + \varphi_{offset}^{C-B} + \delta_{prop}^{C-B} + \tau_{rx}^{C-B} \qquad \text{Eq. 7-25}$$

$$T_{5,k}^{n} = T_{syn,k}^{C} + \varphi_{offset}^{C-n} + \delta_{prop}^{C-n} + \tau_{rx}^{C-n} \qquad \text{Eq. 7-26}$$

With equations Eq. 7-25, Eq. 7-26 and Eq. 7-15 we can get Eq. 7-27. Moreover, the node BNn can calculate ΔT_{k}^{B} using Eq. 7-28. Using these data we obtain φ_{offset}^{C-B} depending on $T_{5,k}^{B}$, $T_{2,k}^{n}$ and ΔT_{k}^{n} (see Eq. 7-29).

$$\varphi_{offset}^{C-B} = T_{5,k}^{B} - T_{5,k}^{n} + \varphi_{offset}^{C-n} - \left[\left(\delta_{prop}^{C-n} - \delta_{prop}^{C-B}\right) + \left(\tau_{rx}^{C-n} - \tau_{rx}^{C-B}\right)\right] \qquad \text{Eq. 7-27}$$

$$\Delta T_{k}^{n} = T_{4,k}^{n} - T_{2,k}^{n} \qquad \text{Eq. 7-28}$$

$$\begin{aligned}\varphi_{offset}^{C-B} = T_{5,k}^{B} - \left(T_{2,k}^{n} + \Delta T_{k}^{n}\right) & \\ - \Big[\left(\delta_{prop}^{C-n} - \delta_{prop}^{C-B} + \delta_{prop}^{B-n} - \delta_{prop}^{B-C}\right) & \\ + \left(\tau_{rx}^{C-n} - \tau_{rx}^{C-B} + \tau_{rx}^{B-n} - \tau_{rx}^{B-C}\right)\Big] &\end{aligned} \qquad \text{Eq. 7-29}$$

Finally, we can estimate φ_{offset}^{C-B} using the Eq. 7-30.

$$\hat{\varphi}_{offset}^{C-n} = \frac{1}{M}\sum_{k=1}^{M}\left(T_{2,k}^{n} - T_{2,k}^{C}\right) \qquad \text{Eq. 7-30}$$

In this way, all sensor nodes are synchronized. There is a secure synchronization between several pair of nodes, which involves a secure synchronization between all nodes inside a group. The next section shows that this process is improved according to number of repetitions.

7.6.1 Test Performance

In order to check our proposal, we have simulated the algorithm proposed in the previous section. We have used Matlab [104] as a simulation tool to show the validity of our proposal. The sensor nodes are randomly placed in an area of 10x10 Km. They have radio coverage of 75 meters, so the distances between nodes, which are communicated, are always less than 75 m. The arrival times follow an exponential distribution. For each offset time, we have considered three cases: a) μ=15 and σ=3, b) μ=20 and σ=6 and c) μ=36 and σ=13. Next figures present the simulation results based on scheme shown in **Figure 7-10**. These graphs show the offset time required between two sensor nodes. The main conclusion from these simulations is the offset time tends to be constant, but now we go to analyze them in detail in the following paragraphs.

Figure 7-11 shows the offset time between sensor node A and B. In this case we can see that when σ is low (between 3 and 6) the offset time is lower than with higher σ. The offset time is around 5 μs with few samples and low σ, and as the number of samples is increasing, the offset time decreases. It is around 1 μs for 40 samples. Simulation behavior is different when we have a σ=13. In this case we have a high offset time between both sensor nodes until 10 samples. Then, its behavior is more stable and it remains constant around 2 μs after 40 samples. These differences are due to variability (σ=13) in the sampling times used for the simulations.

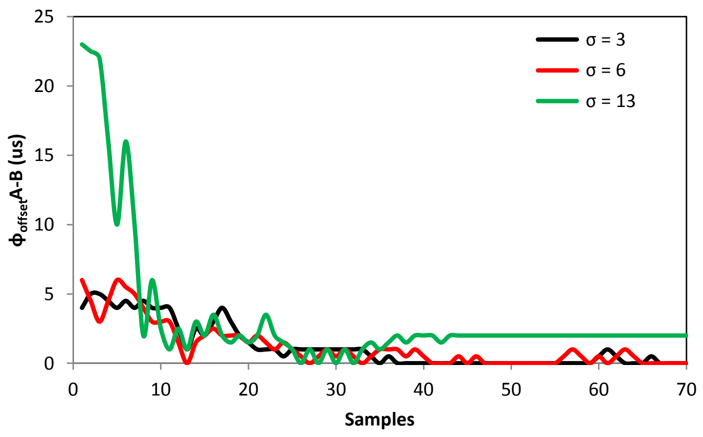


Figure 7-11: Offset time between sensor nodes A and B (φ^{A-B}_{offset} (μs)).

Next, **Figure 7-12** shows us the offset time between sensor node C and n. According to our simulated topology, the most constant values are when our synchronous referenced time has an σ=6. The offset time is around 1 μs in almost all samples. When σ is 3, the offset time is 6 μs in the first samples, but then it is more stable and its fluctuations are between 0 and 2 μs. Finally, the most unstable offset time is when we have a distribution time with σ=13. In this case there are several fluctuations, which vary from 3 to 13 μs until the sample 13. However from this sample, fluctuations are decreasing and they remain between 0 and 4 μs until the 55th sample. From this point, the offset time is stable around 2 μs.

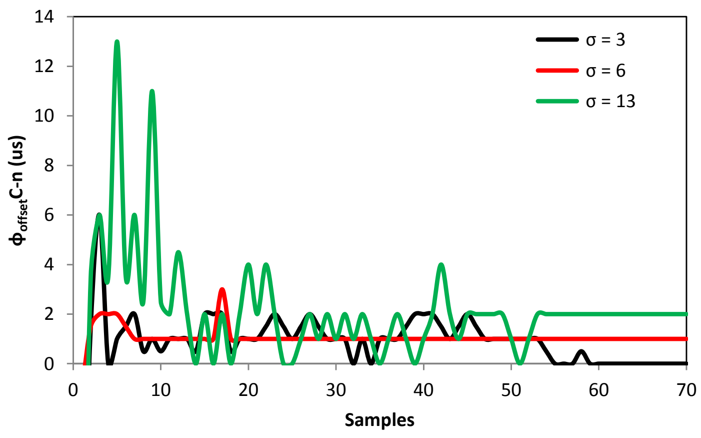


Figure 7-12: Offset time between sensor nodes C and n (φ_{offset}^{C-n} (µs)).

Next figure (**Figure 7-13**) shows the offset time between sensor nodes A and 1 (φ_{offset}^{A-1}). In this case, the most stable result is when σ=3. All samples are around 10µs. For σ=6, the first samples are around 10 µs, but from the 16th sample, it changes to 20 µs. There are several fluctuations between 10 and 20 µs until the 35th sample. Then it remains stable around 20 µs. Finally, the results obtained with σ=13 are quite different on the first samples, the offset time reaches up to 60 µs. Fluctuations are so high until the 40th sample, they vary between 10 and 33 µs indeed. Next, this time is stabilized around 26 µs. The time offset variability in these simulations is the expected one, as our initial distribution has a higher σ, the results are more unstable compared to the results obtained with smaller σ's.

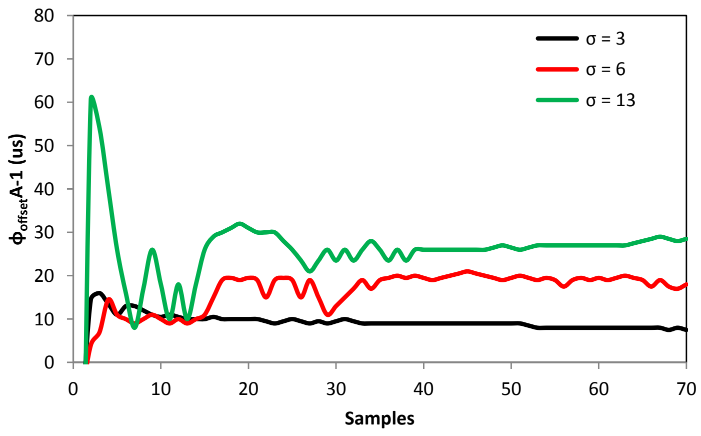


Figure 7-13: Offset time between sensor nodes A and 1 (φ_{offset}^{A-1} (µs)).

Finally, we present the simulations of φ_{offset}^{C-B} (see **Figure 7-14**). Simulations for scenarios with σ=3 and σ=6 are stable from the 15th sample. From this sample the difference between both simulations is 5 µs. When we have σ=3 the offset time is around 10 µs. It changes until 15 µs when we have an arrival time distribution with a σ=6. This offset time increases until 35 µs (it is stable from the 45th sample) when our distribution time has an σ=13. In this simulation the worst case is from 10th to 25th sample. This is due to exponential feature of our arrival time distribution. In this period of time our variance is higher than at other times.

But, the most important feature obtained from all simulations shown (see Fig. 6, Fig. 7, Fig. 8 and Fig. 9), is the stability achieved by our proposal after several samples. So, the proposed system is able to find a constant offset time between each pair of sensor nodes and it never exceeds 35 µs in the worst case. Moreover, the information exchanged between nodes is secure because each message is encrypted using group-based keys.

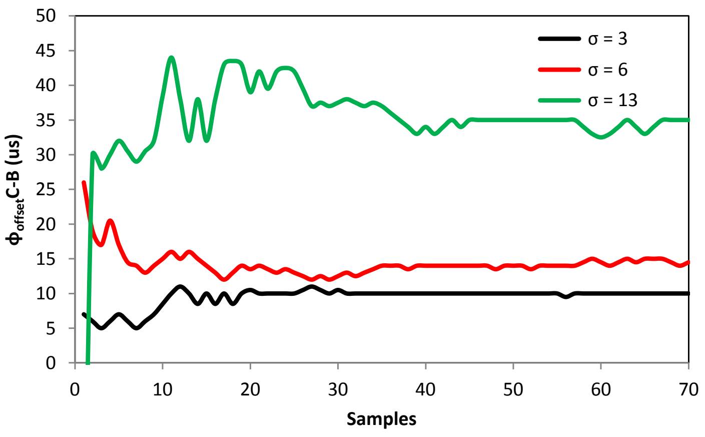


Figure 7-14: Offset time between sensor nodes C and B (φ_{offset}^{C-B} (µs)).

7.7 Conclusion

Firstly, in this chapter we have presented necessities to see that a WSN needs secure communications and time synchronization between sensor nodes. For these reasons, we have shown a simple secure group-based architecture for wireless sensor networks. Our proposal has two security zones. On the one hand we have the intragroup security and, on the other hand, the intergroup security.

Intergroup security of network is based on the symmetrical cryptography with a unique key for all the members of the network. This type of security is used when a sensor wants to send information throughout the network. Intragroup security is made by group keys. These keys are created through pseudorandom functions and they are modified by an update value every time that there is a change in the group. Moreover, we have explained steps needed to exchange these keys between nodes of the same group.

We have estimated the energy consumption given by each operation in this secure protocol for a group-based architecture and we have compared the energy consumption of our protocol when it is used in the elected sensor with other sensor devices.

Then, we have done a review about clock synchronization in wireless sensor network. We have shown the main reasons to use a proper synchronization technique. Besides, the simplest method to make a protocol, which manages the offset times between sensor nodes, have been explained briefly. Lastly, we have also described the typical sources of error.

After this review we have shown our secure group-based WSN and the main issue is the key's exchange. This is an essential part, because it is essential in order to exchange encrypted messages inside a group. Next, we have explained in detail each step needed to know the offset times between sensor nodes. We have based our proposal on receiver-receiver protocol, but adapting the main idea to a group-based topology. All explanation is based on our scheme (see **Figure 7-10**) but it could be developed for any group-based topology. Our proposal has been simulated for the same scenario changing μ and σ values of our time arrival distribution. All cases show that our proposal can resolve the offset time between nodes in any moment. But this offset time is more stable when as the number of samples is going up.

Finally, work presented in this chapter has been published in the following references [26], [108] and [27].

Chapter 8

GROUP-BASED SENSOR NETWORK FOR MANAGING AND CONTROLLING MARINE FISH FARMS

8.1 Introduction

There are several applications' environments where WSNs could be used, as we have seen in Chapter 2 and Chapter 3 of this dissertation. At the end of Chapter 3 we have shown some environments where group-based could function properly. During these years doing this thesis, our research group have taken part in some projects related to marine environment. For instance, our last R&D project accepted by Spanish government (TEC2011-27516) called "Cognitive group-based collaborative sensor network for sensing and monitoring the water environment" is closely linked to this thesis. Its main objective is to research and develop a new sensor wireless network that uses the behavior environment prediction in the aquatic medium by monitoring it. In order to achieve our goal we will apply self-learning algorithms, representation of the knowledge acquired from the environment and the information management of the wireless sensor network.

Aquaculture is an activity directed to the breeding, production and grows of fish, mussels, oysters or other shellfish that live under controlled conditions in aquatic environments. In the fish feeding process, we must make estimations about the amount of food that should be drop into the water. The wasted food could arrive up to 8.26% of the total food. Keeping in mind that the food costs represents around the 60% of total operating costs of the marine fish farm, we should not underestimate the amount of lost food in the fish feeding process [114]. Furthermore, one of the main underwater environmental impacts, in the seabed under the marine fish farm cages and their surroundings, is the deposition of fecal waste generated by the fish.

When we talk about marine fish farms sustainability, we must take into account that this concept focuses its importance on issues related to ecology and the environmental protection. The most well-known factors that affect to the marine fish farm sustainability are:

- Physic-Chemical characterization of the water: Water Temperature, Dissolved oxygen (dO2), pH, Salinity, Conductivity, Ammonium, A Chlorophyll, Total Organic Carbon (TOC), Phosphates, Total Phosphorus (TP), Nitrates, Nitrites, Total nitrogen and Solids suspension.
- Physic-Chemical characterization of the sediments: TOC, soluble phosphorus, soluble total nitrogen and redox potential.
- Granulometric composition of the sediments: Grain-size composition and sedimentology characterization.
- Description of the benthonic communities in the medium: Shanon diversity index, index of fairness, Margalef´s diversity index (Species richness).

Therefore, our main objectives to improve the sustainability in these facilities should be:

- To protect the environment. It is important to reduce the waste deposition and its impact on the ecosystem. Many of them have high nitrate, which could produce eutrofisation effects.
- To quantify the amount of wasted food. It is possible to optimize the amount of food to drop into the cage. The fish food is an extruded food made from raw material that comes from the natural environment. The cost of feed production should be taken into account because it is getting harder to supply the alimentary demands of the aquatic sector.
- We must also take into account the waste and feces of each cage. The droppings of uneaten food form a contamination zone near the cage. The polluted area is related to the type of material that forms the seabed (mud, sand, etc) and distance between the seabed and the bottom of fish cage.

A marine fish farm is typically made by 6 or 12 cages, with diameters ranging from 12 to 50 meters, which can cover an area of approximately 500000 m^2.

Marine farm installation must follow a series of laws that depend on each state or country. Many factors, given by the environment and the marine fish farm characteristics, are taken into account:

- Measures of the depth and the distance from the coast where the marine fish farm is going to be placed.
- Environmental features of the place.
- Occupied/cultivated surface.
- Production system.
- Produced biomass.
- Generated waste materials.

Spain is one of the primary European aquaculture producers in terms of volume [115]. Spanish law is very strict when someone is trying to acquire the approval to set up a marine fish farm. The authorization and its concession are given after a favorable environmental impact study of the place where the marine fish farm is going to be set up. Moreover, marine fish farm managers have to make annual tracking of their activity, where some environmental parameters of the water column and of the sea bed are measured [116] and [117]. Hence, the use of new technologies is needed to solve these issues.

For these reasons, we have seen fit to adapt our group-based architecture to a real environment. In this chapter we propose an underwater wireless group-based sensor network in order to quantify and monitor the accurate amount of uneaten feed and fecal waste deposited on the seabed in order to assure the correct fish feeding process. Moreover, we provide the traffic simulations in order to show the performance of the network for these mobile sensors. We propose this new architecture because current standard protocols are not designed for this purpose and they might not be able to work properly in these cases [118].

8.2 Existing Monitoring Systems

We can find in the literature some simulators to estimate the fish growth and infeed treatment. Others simulate the environmental parameters, e.g. the vertical current shear and its effects on particle dispersion. Many of them only take into account some parameters, but not all. There are applications capable to predict the amount of wastes that can accumulate on the seabed generated by marine-agricultural activities. A well-known simulator is DEPOMOD [119].

Some research centres have developed applications with the main purpose of measuring the sustainability in the fish farm. Some of them are:

- AQCESS: Aquaculture and Coastal, Economic and Social Sustainability.
- BIOFAQs: Biofiltration and Aquaculture: an Evaluation of Substrate Deployment Performance with Mariculture Developments.
- ECASA: Ecosystem approach for sustainable Aquaculture.
- ICES: Working Group on Environmental Interactions of Mariculture.
- MERAMED: Development of monitoring guidelines and modeling tools for environmental effects from Mediterranean aquaculture.
- MEDVEG: Effects of nutrient release from Mediterranean fish farms on benthic vegetation in coastal ecosystems.

In order to know the environmental impact of the marine fish, A. Stigebrandt et al. developed a model for estimating the holding capacity of the sites for fish farming in [120]. It is based on four sub-models. Expressed in terms of maximum fish production per month, the holding capacity is estimated with regard to three basic environmental requirements: a) the benthic fauna at a farm site must not be allowed to disappear due to accumulation of organic material, b) the water quality in the net pens must be kept high and c) the water quality in the areas surrounding the farm must not deteriorate. The model provides a structured and quantified description of significant environmental aspects of fish farming.

Soonhee Han et al. designed and implemented an aquaculture environment monitoring system in [121], which monitors the aquaculture environment and sends an alarm to the aquaculture farmers when an inadequate environmental situation is detected. This system is centralized and moreover the nodes are connected to the exterior with a cable, so it does not present any big challenge and it is not scalable.

Underwater wireless sensor networks allow several types of architectures [36], which may be helpful to monitor the seabed and solve the aforementioned issues. Moreover, wireless sensor networks have a large number of features and provide many benefits to monitor and control any type of environment [33]. One of the main issues that a wireless sensor network designer should pay attention is the how sensors are located and placed in the environment [122], the other one is their radio coverage and their sensing coverage area [123].

As far as we know, there is not any underwater wireless sensor network deployed for monitoring and controlling the sediment depositions caused by marine fish farms. Moreover, we have not found any proposal of an underwater wireless group-based sensor network in the related literature.

8.3 Problem description and motivation

With the assembly of the marine fish farm cages, soon an important external source of organic matter will appear. This organic matter is generated by the remains of fish food, lost through the holes in the nets of the cages and the fish fecal pellets. The wastage accumulation in the seabed causes notorious changes in the silt chemistry of the nearby farm areas [124]. This area could be several hectares long. Even in sandy seabed and on the surface of seaweed, a specific analysis of some parameters, such as prairie density, is necessary.

Another important factor to keep in mind is personnel security in charge of the feeding process. Currently, the feeding process is carried out distributing the food by hand or impelled canyons by air. Also, crafts with fixed pipes in each cage impelled by air compressors can be used or by self-demand troughs. Additionally, the control of the exact moment when the food begins to fall to the seabed is performed by scuba-divers or by some submarine cameras placed at the bottom of the cage. They give warning of the moment when the food begins to leave the cage. The fish must be fed all year round, and the conditions of the water are not always good and suitable for immersions. Moreover, both scuba-divers and submarine cameras (and their maintenance) have associated high economic costs.

Systems based on mathematical formulations, estimate the quantity of necessary food in the tank by knowing the number of fish in the cage and supposing that all the fish in the cage have the same weight and size. Although, statistically many of them will be similar (the fish sizes are distributed according to a Gauss bell) there will always be a group that will not be similar (e.g. dominant individuals usually exist and eat more than others) and probably this group will be infra or overfed. Some of them could neither eat so fast nor at the same time as the dominant ones. Therefore a certain percentage of the fish might not be fed correctly because the food has been moved away too soon or, otherwise, the whole group of fish could not eat the foreseen quantities and the food will be wasted, so the environment would be contaminated. An appropriate sensor system would avoid some deaths from bad feeding.

Environmental impact is very difficult to evaluate from an economic point of view, but it is necessary to point out the quantity of food that gets lost. Generally, it varies according to the feeding strategy of the fish farm (usually based on tables, until they satiate). Some published works show that the loss can reach percentages of over 10% of total food [114]. Considering the reports of unitary costs of equipment and personnel in marine farms [125], the total annual cost of food for a farm of six cages with a 25 m diameter is approximately 3908000 € and this, would suppose losses, only in food, for the installation approximately of 390800 €. The need for a monitoring and control sensor system for fish feeding in marine fish farms is justified. An adequate feeding distribution system is essential to assure a homogeneous growth of the fish cultivation [126].

8.4 Seabed depositions estimation for sensor placement

One of the main issues to take into account in sustainability in marine fish farms is how fecal pellets and uneaten feeds are spread under the cages. They are settled some distance from the farm implying an adversely impact in the coastal environment. In this section we study all issues that may affect the placement of the sensors on our underwater wireless sensor network. We study the deposition in terms of studied dispersions models, current speeds and settling speed as a function of the particle features.

There are many studies about the factors influencing the sedimentation and accumulation of organic material under and near the fish cages [127], but we are interested on specifying their distribution area, in order to determine the best sensor distribution along the seabed surface. Many factors can be taken into account to measure the waste deposition: biomass of the fish, metabolic rates of the fish, settling rates of excess fish feed, settling rates of fecal pellets, feeding rates, amount of excess (waste) feed, the consumption of waste feed by other species, rate of decay of organic particles on the bottom, sinking velocity of the particles, velocity and direction of the current, depth-varying currents and the water depth.

The physic-chemical characterization of the water could be used to estimate the pollution level in an area. Thus, it is possible to estimate the concentration or the amount of different elements like carbon, nitrates and phosphates generated from wasted food and feces. There are several studies and works that provide some expressions. Stigebrandt et al. present a dispersion model [120] that gives the mean carbon emission from the net pens. Joining several formulas of these authors, we obtain Eq. 8-1.

$$F\left(\frac{g}{m^2 \cdot Day}\right) = \frac{1}{2} \cdot \frac{T_P}{\Delta f}(FCR - FCR_t + 0.1) \qquad \text{Eq. 8-1}$$

Where, *FCRt* is the theoretical feed conversion ratio; *FCR* is defined as the factual feed conversion ratio; T_P is the Total Proteins and Δf is the total area of the cage. In [120], authors estimated that FCR−FCRt=0.3, so the waste feed equals 0.3 kg per each kg of produced fish. Assuming the T_p can be a value between 20 and 30 (depending of the fish species) [128], and the fact that there are different cage sizes. We can estimate the carbon that is being released into the environment where the cage is located. **Figure 8-1** shows these estimations.

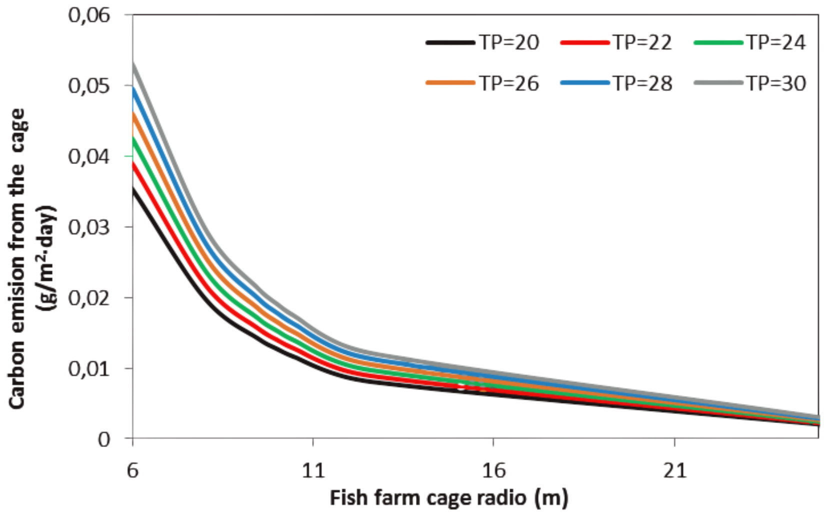


Figure 8-1: Carbon emission from the cage.

We can estimate what are the most affected surrounding areas of the farm considering several parameters such as the depth under the cage (d), the mean current speed (V), the current direction (Ө), the settling speed (u) and the position of each cage (x, y). Many of these parameters, such as sea current speed, depend mainly on the fish farm location, but there could also register variations depending on the season. **Table 8-1** shows different values of current speeds for different locations in the world. Another parameter affects to the seabed deposition is the settling speed of the particles.

Table 8-1: Current Speed.

Reference	*Placement*	*Current Speed mean (m/s)*
[27]	North Balearic Thermal Front	7 cm/s
[27]	Eastern basin around Crete Island	8 cm/s
[28]	Coast of British Columbia	6.7 m/s
[29]	California Currents and the Canary Current	0.03 to 0.07 m/s.
[29]	The Gulf Stream, and the Kuroshio Currents	0.4 to 1.2 m/s.

Table 8-2 shows different settling speeds as a function of the particle features and the collection method [129].

Table 8-2: Settling Speed Particles.

Specie (size in mm)	*Particle settling velocity (mean) (cm/s)*	*Collection Method*
Seriola quinqeradiata (n/a)	5	n/a
Salmo salar (n/a)	1,7 - 6,0	n/a
Salmo salar (n/a)	2	In situ trap
Salmo salar (n/a)	3,2	Siphoning
Salmo salar (>2 mm)	2,9	From aquarium tank outlet
Salmo salar (4.0 mm)	5,4	Stripping
Dicentrarchus labrax (n/a)	0,64	Polythene trap
Dicentrarchus labrax and Sparus aurata (0.3–6.2 mm)	0,7	Polythene trap

Taking the measurements gathered in reference [120], we can estimate analytically the sedimentation *S* as a function of the distance, for a sinking speed per sinking time equal to 10 (δT=10). This estimation is shown in Eq. 8-2 (it has a correlation coefficient of 0.9985).

$$S = -9 \cdot 10^{-7} r^4 + 7 \cdot 10^{-5} r^3 - 0.0014 r^2 + 0{,}0018 r + 0{,}1749 \qquad \text{Eq. 8-2}$$

Figure 8-2 shows the depositions of a marine fish farm with 6 cages, each one with 15 meters of radius, that are placed in the positions (20, 20), (60, 20), (100, 20), (60, 30), (60, 70) and (60, 110) for δT=10.

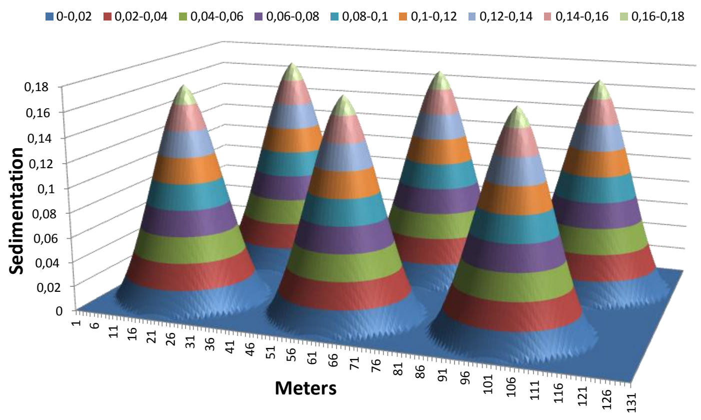


Figure 8-2: Sedimentation for *δT*=10, in a fish farm with 6 cages.

On the other hand, we also estimated analytically the sedimentation *S* for a sinking speed per sinking time equal to 5 (*δT*=5). This estimation is shown in Eq. 8-3 (it has a correlation coefficient of 0.9995).

$$S = 10^{-8}r^5 - 2 \cdot 10^{-5}r^4 + 8 \cdot 10^{-4}r^3 - 0.01r^2 + 11 \cdot 10^{-4}r + 0.53 \qquad \text{Eq. 8-3}$$

Figure 8-3 shows the depositions of a marine fish farm with 6 cages, each one with 15 meters of radius, that are placed in the positions (20, 20), (60, 20), (100, 20), (60, 30), (60, 70) and (60, 110) for *δT*=5.

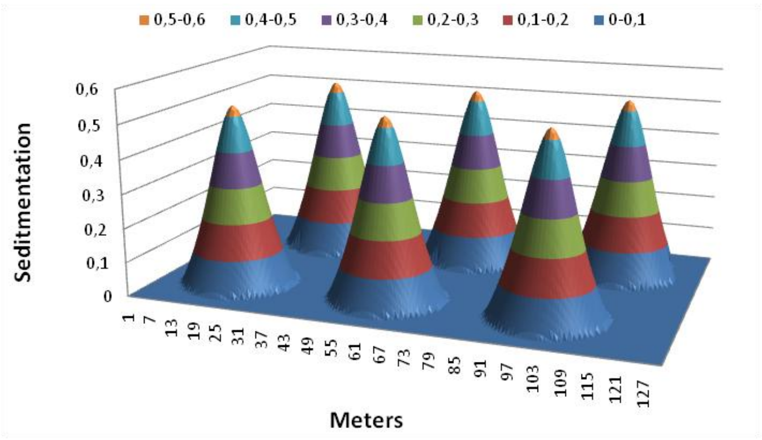


Figure 8-3: Sedimentation for *δT*=5, in a fish farm with 6 cages.

We have observed several differences between both cases. On one hand the peak of sedimentation is higher in the center of each cage for *δT*=5 than for *δT*=10. On the second hand, the separation between the deposition zones is higher for *δT*=5 than for *δT*=10. The study presented in this section is later used to decide the number of sensors and their most appropriate location.

8.5 Sensor network proposal

To design our sensor network, we have studied the types of underwater sensors available in the market that can be useful for water and fish monitoring. They can be used to measure the following [118]:

- Speed and direction of the water current.
- Parameters of water quality as pH, temperature, salinity, etc.
- Surge, wave height, wave frequency.
- Water turbidity, suspension particles.
- Seismic activity.
- Underwater displacements.
- Matter reflections inside the water (using acoustic sensors).

The fish behavior is highly related to the aquatic medium where they are placed. Our proposal is based on the idea of measuring the behavior of the fishes inside the marine fish cages. Then, we can predict if they are hungry or satisfied. To achieve this goal, we have to mix some data gathered from different types of sensors placed in the marine fish cages. We must take into account that some parameters may be affected by the behavior of the fishes because they are hungry, or otherwise, they could be affected in the same manner for other reasons (such as depredators presence) or otherwise, different fish species exhibit different behavior, thus giving different values for these parameters [130]. Moreover, fish behavior also depends on the fish cage dimensions [131]. There are other parameters that do not affect fish feeding, such as the water turbidity.

The first step is to know which parameters should be sensed to provide us the information needed to know if the fishes should be fed or not. This study has been made by reading all the known works on fish behavior while they are being fed. Some studies demonstrate that temperature affects both the breeding and the feeding process [132]. Moreover, some fishes do not eat when the water temperature is lower than a defined value. When a fish is full, it tries to feed fewer times and wastes more time moving to get the food [133]. But, there are some species where this relationship is not too strong (such as the SalmoSalar) [130]. So, this information has to be mixed with other parameters to guess the right behavior. On the other hand, fishes try to come closer to the water surface when they are hungry because it is the place where they are fed, and they sink to the bottom of the cage when they are satisfied.

Many studies demonstrate that the oxygen in the water varies during the feeding process [134], so this parameter should be measured in our sensor network. Moreover, oxygen is a critical parameter, because very small reduction in the oxygen level affects the fish growth and appetite, and its lack called hypoxia would be the cause for the death of many fishes.

Finally, some external parameters should also be measured. When the thermocline is too high, the fishes do not eat the foreseen quantities. Moreover, when there is wind or lot of current, the fish swim very quickly, and the food can go through the sides of the cage.

Our system takes as a starting point the type of fish that is going to be fed, for this reason we should choose the correct food and the feeding hours. The food dispensation system receives orders from the management system about the food quantity dispensed for time unit that should fall in the cage. This food quantity dispensed for time unit is controlled using the information taken from the distributed sensor data fusion algorithm described in a later section. This control is carried out in real time.

Bearing in mind the information obtained from the related literature, we have designed a sensor network (shown in **Figure 8-4**) for the fish cage that is formed by:

- A group of sensors that senses the water column temperature.
- A group of sensors that senses the water column oxygen.
- A group of sensors that measures the fish displacement speed.
- A group of sensors that estimates the fish biomass distribution placement in each moment.
- An acoustic sensor that lets us know when the food begins to fall in the seabed and its quantity.
- A sensor network placed on the seabed to control sustainability (amount of faeces and feed wasted).
- Outside sensors. They allow us to measure whether there are depredators outside the cage, to know if the fishes are stressed because of it and the state of the water current, in order to tune the fish speed.

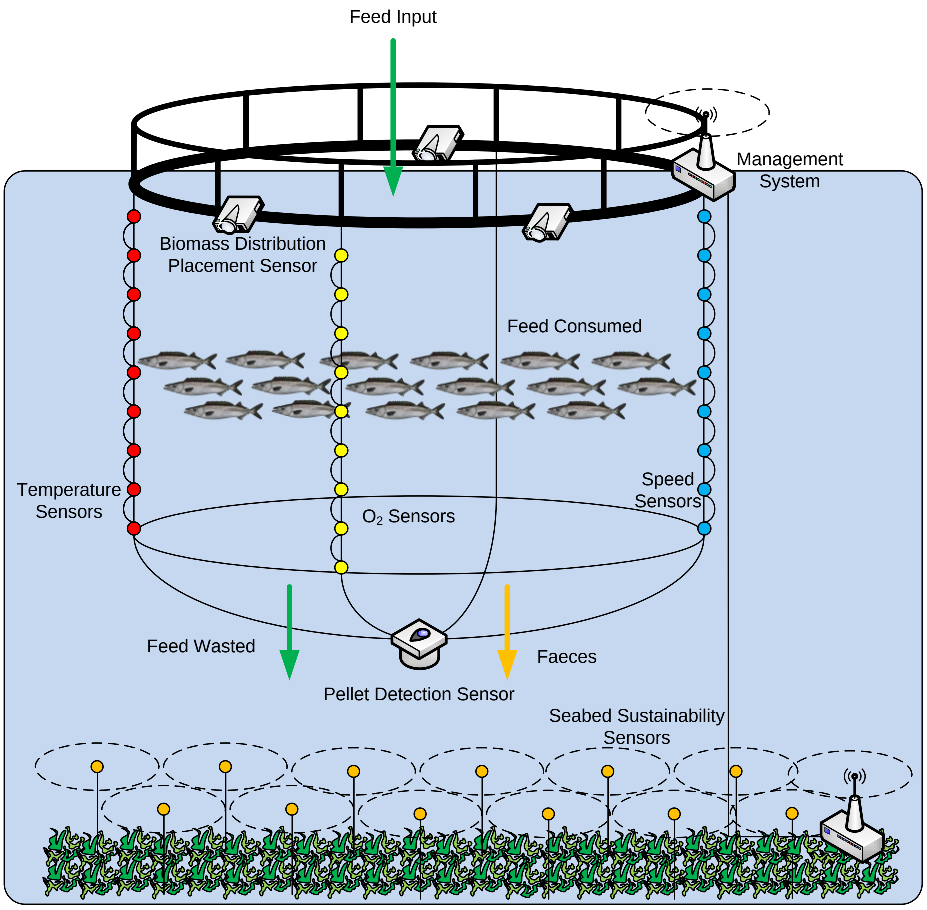


Figure 8-4: System for marine fish farms.

8.5.1 Temperature sensors

According to some studies, each species grows best between some specific temperature ranges. Low temperatures will affect the feeding process because the fishes eat less. To control the water column temperature in the marine fish cage, we have introduced temperature sensors along the side of the cage. These sensors are placed at one meter interval. Each sensor gives us the value of the water temperature at a certain depth. All temperature sensors are connected by serial connections to the management system. The temperature value is given unequivocally using a sensor identifier (sID). This information is received by the management system, which will process the data.

8.5.2 Oxygen sensors

Oxygen is a limiting factor in fish growth. Fishes do not feed well at low oxygen concentration. Each species has its optimum oxygen range. We have placed oxygen sensors over the side of the cage every meter. The oxygen level does not change much in each meter, but this will allow us to have a detailed view of the oxygen where the fishes are placed. The oxygen sensors are connected to the management system through serial connections. Each sensor node gives the O2 measured value and its identifier. The information is sent to a management system, which will process the data.

8.5.3 Displacement speed sensors

In many species, the speed of the fish is directly related to their hunger. If the fishes are moving quickly, it is because they are hungry (but it could also be caused by the appearance of a depredator near the cage). To measure the speed of the fishes, we have selected an underwater sensor that senses the water stream produced by the movement of the fishes. The speed sensors are along the side of the cage. The sensors are placed every meter.

These sensors are calibrated in advance because we know that there are water currents that could cause false alarms. Moreover, the data will be compared with water current sensors placed outside the cage (explained later). Each node has an identifier and is connected through a serial interface. Each node sends the speed measured and its sID to the management system, which will process all data.

8.5.4 Biomass distribution placement sensors

The distance from the water surface where the fishes are placed is an important parameter in the feeding process. The fish tend to go to the water surface when they are hungry and once their appetite is satisfied they go down to the bottom of the cage. It is therefore important to locate the fish inside the cage. In order to know the placement of fish we opted for the use of three acoustic sensors located at the top ring of the cage, towards the deep. These three acoustic sensors let us know the distance of the shoal through the reflected signal.

These sensors are also connected via serial connections with the management system. The data is sent jointly with their sID in order to estimate the placement of the shoal.

8.5.5 Seabed sustainability sensors

One of the main issues to take into account in sustainability in marine fish farms is how fecal pellets and uneaten feeds are spread under the cages. They are settled some distance from the farm implying an adversely impact in the coastal environment.

In order to control seabed sustainability we have used two types of sensors. First, a sensor placed at the bottom of each cage, which allows knowing when food begins to fall down. And then, a group-based sensor network strategically located where there may be more sediment. This underwater sensor network will use a wireless channel to communicate every sensor node with its nearest one. All data will be sent using other sensor nodes until information arrives to a sink node.

8.5.6 Outside sensors

- Presence sensors: To detect predators around the marine fish cages, our system will use several underwater video cameras located outside the cages. These video cameras will be located in strategic positions properly oriented to see the largest amount of space with the least possible number of video cameras. Image-processing software will warn us of the presence of fishes outside the cages. Moreover, it can also be used to know if the fishes are leaving the cage because the net has been broken.
- Water current sensors: The use of multiple sensors placed outside the cages to measure water current is necessary to calibrate the measure taken by the speed sensors placed inside. The innovation of our system is reflected in the fusion of data between the speed sensors placed inside the cages and current sensors placed outside. The water current sensors will be placed in the same direction of the speed sensors in order to provide the most accurate measurements.

8.6 Group-based seabed network

In order to design a group-based underwater wireless sensor network, it is necessary to make an accurate design and planning of the system [135] in order to obtain the required measurements. So it should be kept in mind several factors such as:

- The number of sensors
- The distribution of sensors
- Sensor wireless coverage
- Sensed coverage
- The sensors placement
- Periodicity taking samples

These factors are not fixed because they depend on the number of cages, their size and the environment where they are placed. In our case, the main problem is the placement of the sensors. If we place the sensors stuck to the seabed, we will need many sensors to cover such area. Thus, we have decided to use ultrasonic sensors placed at 20 cm. from the seabed, with their beam directed downward in order to cover bigger area. Ultrasonic sensors are used to determine the distances to possible obstacles and to monitor a space. These sensors can control larger areas than other sensors such as capacitive or inductive sensors.

Ultrasonic sensors can cover a distance of about 0.25 m. to 6 m., depending on the used model. **Figure 8-5** shows the seabed network proposal.

8.6.1 Group-based topology

Group-based topologies improve the performance of the whole network by significantly decreasing the communication cost between end-nodes [136]. We have used group based architectures in WSNs obtaining excellent results [137]. Moreover, we have proved that cooperative group-based wireless sensor networks to save energy and improve communications [58].

Our group-based protocol operates above the transport layer and it is therefore independent of the protocols used in lower layers such as medium access control and routing. This subsection explains the group-based network topology and the role of the nodes in the network. In order to make our proposal easy to understand, we use a fish farm composed by 6 cages. It could be easily extended due to its simplicity. A scheme of our network topology is represented in **Figure 8-5**. It also shows a simple distribution of sensor nodes and their function.

The network topology is divided into several groups, a group per cage. We differentiate several zones in each cage (there are more nodes than the points shown in the figure, but we have only drawn few to clear it). As we have discussed before, there are more sensor nodes (higher density) in the center of each cage because in this area there are higher levels of sediment. This density decreases when the distance to the center of the cage increases. There are three types of nodes in our network:

- Regular sensor nodes (RS): These nodes are able to sense the sedimentation of a cage and to communicate with other sensor nodes in order to send collected data. They are represented in **Figure 8-5** as red circles.
- Border sensor nodes (BS): These nodes perform the same tasks than the regular sensor nodes, but they also send information from the nodes in their same group to another or to a sink node. In **Figure 8-5**, they are green circles.
- Sink sensor nodes (SS): These nodes collect the information sensed by all nodes. This information is sent to the control center, which processes the data and creates alarms in case of higher values than a threshold. They are shown in **Figure 8-5** as purple circles.

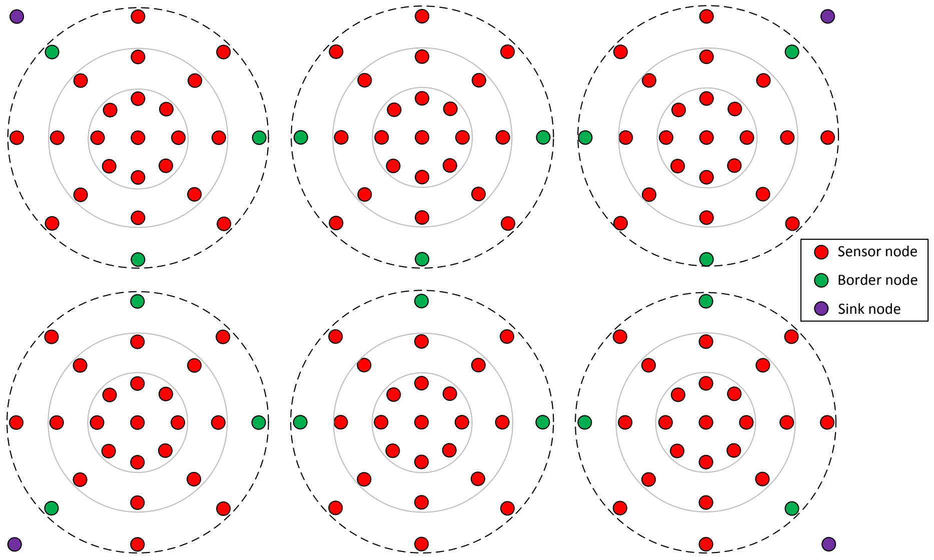


Figure 8-5: Underwater Group-based WSN for marine fish farms.

In order to have a dynamic network with nodes' discovery, we have planned two different types of role procedures. The first role procedure is dynamic, and depending on the network formation, the node can be RS or BS. The second procedure happens for a fixed role: Sink sensor nodes. They will be configured before its installation undersea. Each SS will know its position in advance because these types of nodes receive the data gathered from the group-based network. Sink nodes are known nodes and they are located in strategic places. For example, in our topology (see **Figure 8-5**) they are placed in the corners.

Procedure for non-SS nodes is as follows. They are placed from the center of each cage to the border. When it joins the network, it sends a *Hello* message in order to see if there are more nodes in the network. If it does not receive any response during a predefined period of time, it will create a new group and it will become a RS node. But, otherwise, if it receives a response, it will become RS node and it will create an adjacency with the node that has replied its *Hello* message. In case of multiple replies, even from different groups, new node will process them and select the best group to join in. If all neighbors' replies come from the same group, next step is to estimate which is the farthest node.

This estimation is used to decide the size of the group. Group's size is limited by the *L* variable. This parameter is set in each node before its underwater installation. The farthest distance is compared with L parameter. If the distance is less than *L*, the node will become BS. But, if this distance is greater than *L*, the node will create a new group and then it will become RS. Finally, when a node receives several replies from several groups, it must choose the best group to join. This selection process is explained in [20] (authored by the same authors of this paper). In this selection process, a node chooses a group based on several parameters, but always the farthest distance must be taken into account in order to carry out the specification of the distance *L*. When the node selects the best group it belongs to that group. When a node is a BS and it is connected with another BS, it sends this information to it nodes of its groups in order to let them know it. **Figure 8-6** shows the algorithm explained.

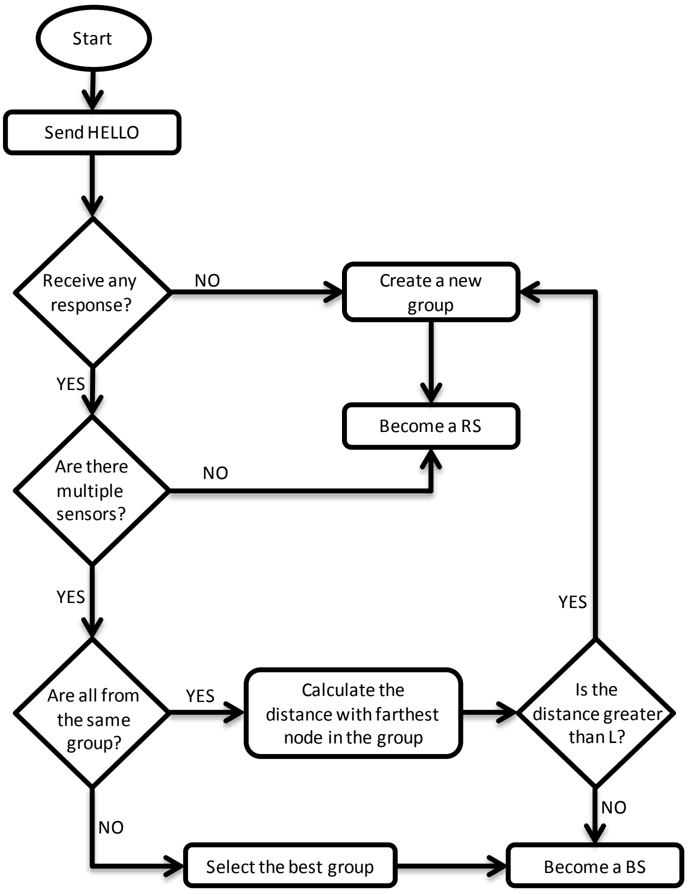

Start
Send HELLO
Receive any response?
NO
Create a new group
YES
Become a RS
Are there multiple sensors?
NO
YES
YES
Are all from the same group?
YES
Calculate the distance with farthest node in the group
Is the distance greater than L?
NO
NO
Select the best group
Become a BS

Figure 8-6: Node role selection process.

8.6.2 Inter-Group and Intra-Group Communication

Each RS sends a Hello message (with its identifier and the sediment level) to its neighbors. Each neighbor replies with a *HelloReply*, which includes the same information of the neighboring nodes. This connection is ended with an *ACK* message. The delay between border nodes is stored, because it is needed in order to know how to reach the nearest SS. In our network the closest node is measured in terms of delay and number of hops.Intra-group communication is performed between BSs. BSs exchange information between groups using *GroupInfo* messages. BSs acknowledge the correct reception of *GroupInfo* messages with acks. *ACK* message include groupID, neighboring groups and the

BSs that connects to each neighboring group. Finally, BSs send the *GroupNeighbor* message to notify which are their neighboring groups and/or its connection to the sink node. **Figure 8-7** shows the exchange of messages between neighboring nodes in same group and between BSs of two different groups.

Sink nodes are known nodes and they are located in strategic places. For example, in our topology (see **Figure 8-5**) they are placed on corners. When a sink node is set up, it sends a *SinkNodeActivated* broadcast message in order to announce it. This message includes the identifier of the connected group. Although in our topology example there are 4 sink nodes. Only 2 of them work at the same time, one in each part of the network (we have place 4 for failure-tolerant purposes). The other two will be in sleep mode and if any sink node fails, one of them will be activated. Each sink node is programmed to send a *SinkNodeFail* message when its battery level reaches 4%. This message will be sent through the groups to all BSs connected to sink nodes. The closest sink node to the failure sink node will be elected as a new sink node, and then it sends a *SinkNodeActivated* message. Each sensor node in the network will change its sink node when it receives a *SinkUpdate* message or when the neighbor group detects the failure of a sink node. The role discovery process can be viewed in **Figure 8-8**.

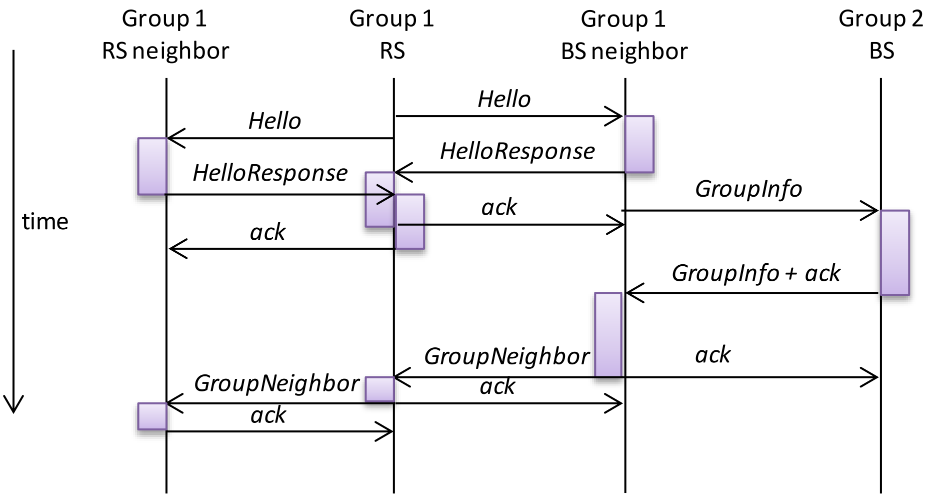


Figure 8-7: Exchange of messages between RSs and BSs.

Data collected by each sensor node will be sent to the BS which is closest to its sink node. Each group will know the most suitable sink node due to the use of the information about *GroupInfo*, *GroupNeighbor* stored in SSs and BSs and the *SinkNodeActivated* messages. The information will be routed among different groups using the furthest nodes from the center of each group and the shortest delay path (more information about group-based routing procedure is explained in [20]).

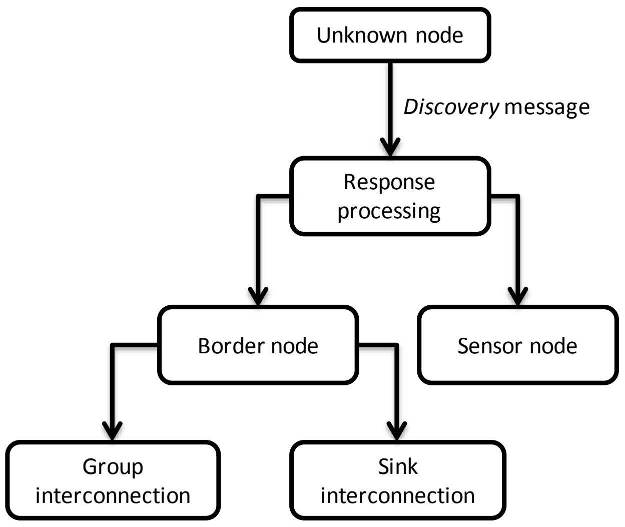


Figure 8-8: Role discovery process.

In order to avoid the continue disconnections and reconnections we have developed a prevention based on timers. Sensor nodes are moving continuously due to: a) movement of the sensor nodes, b) extreme conditions of underwater channel, and other reasons seen in previous sections.

As we know, the solutions based on timers are used in several routing protocols such as RIP, OSPF, IS-IS, etc. For this reason, they are a good solution to avoid flip-flops and having the routing table updated constantly. They prevent us to exchange excessive control messages. In our group-based network, because of the sensor movement there will be many joining and leavings neighbor's table entries. In order to avoid the problem of sending a wrong routing table to the neighbors, we have developed a mechanism based on timers. The mechanism does not consume too much energy because they are only used reactively. We assume that the probability of finding unavailable neighbors when a sensor node sends some information is very low because when they sensors are moving, they generally move in the same direction (following the water flow). Hence, next procedure is only carried out when a sensor node tries to send some information and it does not find neighbors to send this information. We have defined three types of timers: *Hello_timer, Hold_timer* and *Clean_timer*. The protocol procedure using timers is described in our previous conference paper [28].

8.6.3 Simulations of group-based seabed network

This section explains our simulation test and shows the performance of our proposed group-based protocol.

8.6.3.1 *Test bench*

In order to evaluate the proposed wireless group-based network, we varied the number of sensor nodes and the number of groups. We have used OPNET Modeler network simulator [83]. We simulated two scenarios, one with 68 sensor nodes (6 cages with 11 sensor nodes plus 2 working sink nodes) and another with 200 sensor nodes (6 cages with 33 sensor nodes plus 2 sink nodes). Both scenarios had 2 working sink nodes that were placed in the opposite corners of a 500x500 m2 area. This area simulates a real seabed of a fish farm.

We studied several parameters in order to see how performs our system. In these simulations we have a physical channel similar to the aquatic channel. Nodes used in our simulation were MANET nodes, but with very limited capabilities. They have a very low processing rate and transmission rate. This is due mainly to two reasons: a) low processing rate because we need to save energy and b) a low transmission rate due to the aquatic environment. Sink node has more processing capacities than the rest of the nodes. Because it receives the traffic from all nodes, it needs more processing rate. Moreover, it is connected to a buoy with a cable, for this reason it does not have power restrictions. Nodes inside each group use AODV as routing protocol. Our sink node is connected to a buoy, which is directly connected to a control center.

Nodes can move randomly during the simulation only around 50 cm, because they are fixed to the seabed with a wire. The obtained data neither depend on the initial topology of the nodes nor on their movement pattern. The traffic load used in the simulations is MANET traffic generated by OPNET. We inject this traffic 100 seconds after the beginning. The traffic follows a Bernoulli distribution. The packet size follows an exponential distribution with a mean value of 128 bits. The injected traffic is started on a random sensor node, obtaining a simulation independent of the traffic source, but all traffic go to the sink node.

8.6.3.2 *Network simulation with 68 nodes*

This subsection shows wireless group-based sensor network performance when there are only 68 nodes. In **Figure 8-9** we see the traffic sent by a regular sensor node to the network, which emulates a sensor sending continuously the sensed data from the seabed. That traffic is around 8 kbps, which is an acceptable bit rate for underwater communications. Therefore, this simulation shows that traffic generated by our system is acceptable for underwater acoustic communications. We can observe that traffic sent to the network is quite stable.

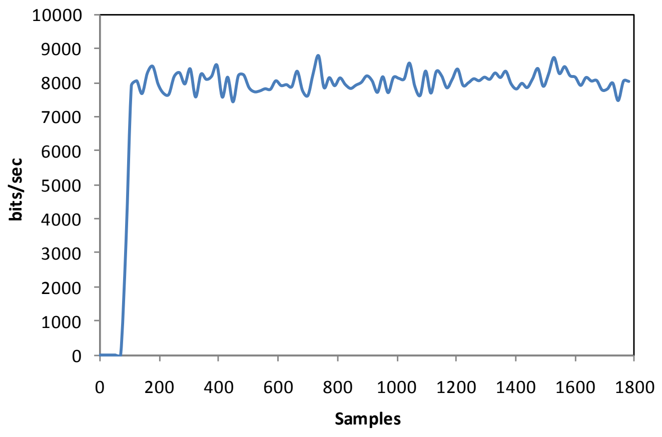


Figure 8-9: Traffic sent to the network by a sensor node.

In **Figure 8-10** the instantaneous network delay is shown. The average delay is around 10 seconds. There are several peaks higher than 18 seconds. This delay is acceptable because our system does not require real-time process. The main cause of this delay is given by the forwarding process of the nodes when they are transmitting the information in the underwater environment. In order to have high efficiency in the transmission, we must wait for a while between the reception and the transmission in order to avoid the interferences of the echoes given by the multipath effect.

Figure 8-11 shows the traffic being delivered through the network. The 68 sensor nodes scenario causes an estimated traffic of 650 kbps. This traffic is quite stable due mainly to the stability of the traffic sensed by the sensor nodes. We observed that the traffic does not exceed 700 kbps in any case. In the worst case (the traffic peak is around 700 kbps), the sensor node was sending around 10.9 kbps, an acceptable bit rate for a regular wireless sensor nodes in the underwater physical channel.

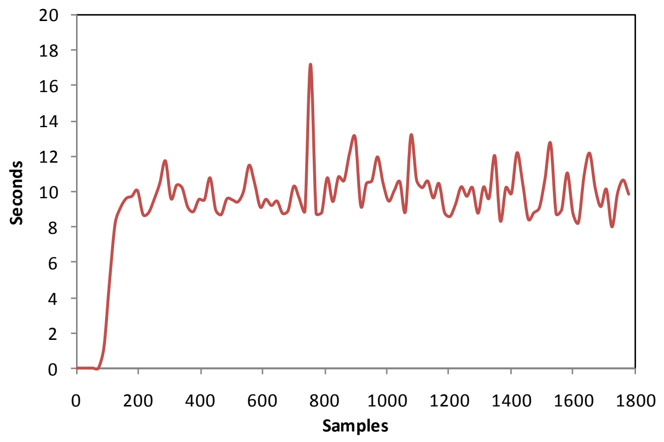


Figure 8-10: Communication delay.

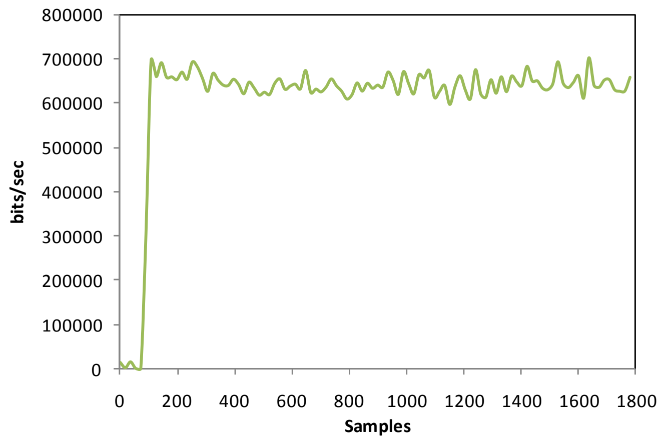


Figure 8-11: Load of the network.

We have also measured the amount of data dropped because MAC data link layer limitation. That is, when the data to be sent is higher than the MAC maximum transmission. In **Figure 8-12** we observe that the average amount of data dropped is around 35 kbps in the worst case. This means that lower layers drop the 5% of the traffic when the network is overloaded, so upper communication layers must recover or request the missing data. This performance test shows the difficulty of transmitting data when we have a very limited communication channel.

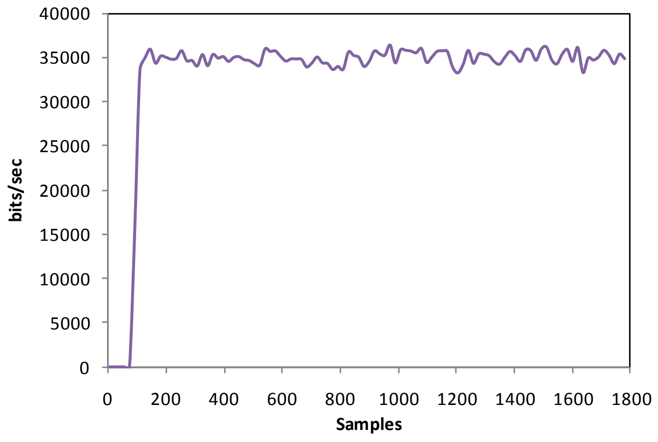


Figure 8-12: Data dropped in an overload case.

Finally, in order to see the retransmission attempts needed to recover the data dropped or lost we provide **Figure 8-13**. In this case, we observe that 2.5 packets are needed to have a good communication between a sensor node and the sink node. In an error-free communication, this figure would have zero values in all samples. When there are losses (mainly given in shared mediums), retransmissions are necessary, so destination should request missing information.

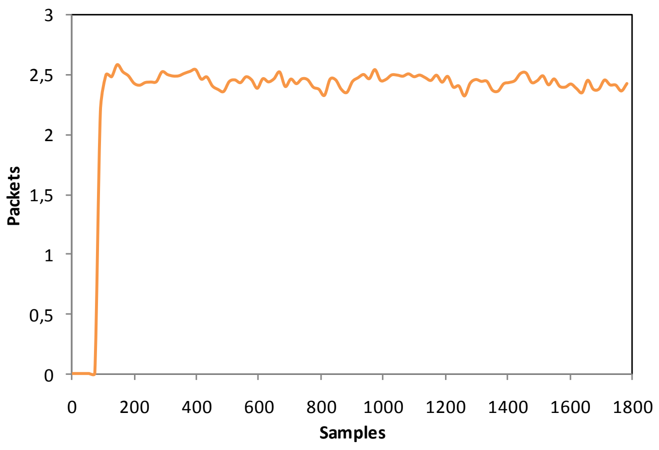


Figure 8-13: Retransmissions attempts.

8.6.3.3 *Network simulation with 200 nodes*

In this subsection we measured a scenario with 200 nodes grouped in 6 groups and compared it with the results obtained in a network without groups. **Figure 8-14** shows the traffic sent to the network by the sensor nodes, which simulates the transmission of the sensed data from the seabed. The traffic sent to the network is around 25 kbps regardless of the number of groups in our architecture, which is an acceptable bit rate for underwater acoustic communications. We can also observe that it is quite stable, which is good for the network performance.

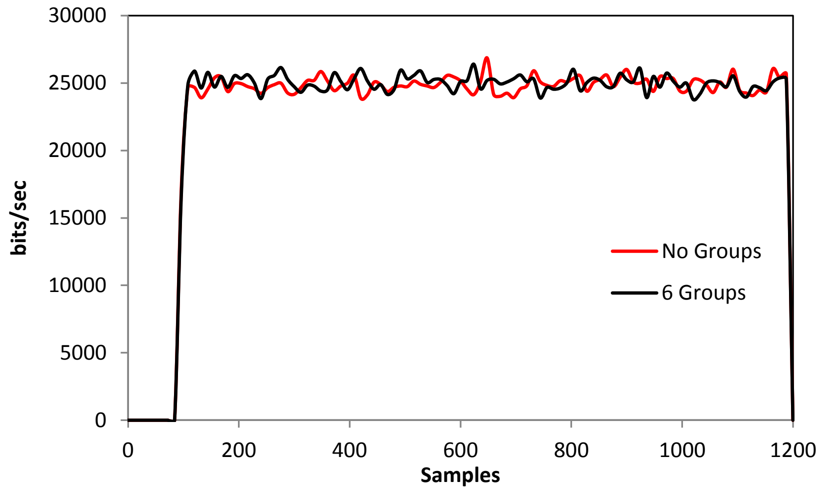


Figure 8-14: Traffic sent to the network because of the sensed data.

Figure 8-15 shows the amount of traffic sent to the network. When there are no groups, there is an estimated traffic of 7 Mbps. We observed that this traffic decreases when the number of groups increases. When there are 6 groups there is an average of 320 Kbps. In the worst case, the traffic does not exceed 650 kbps. In that case, sensor nodes do not send more than 3.25 kbps, an acceptable bitrate for underwater acoustic communication.

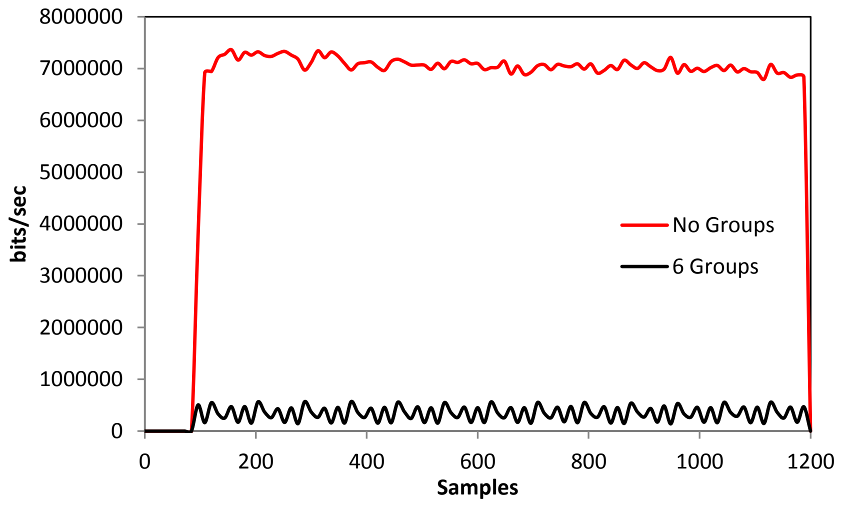


Figure 8-15: Load of the network.

Finally, in order to see the retransmission attempts needed to recover the data dropped by lower communication layers when the network is overloaded, we gathered the measurements shown in **Figure 8-16**. In this case, we can observe that there is an average value of 2 packets when there are no groups, but this value decreases to 1 packet when there are 6 groups. That is, the network only needs 1 retransmission attempt to recover the data dropped and have a correct communication between any sensor node and the sink node. In an error-free communication, there will be zero packets in the graph.

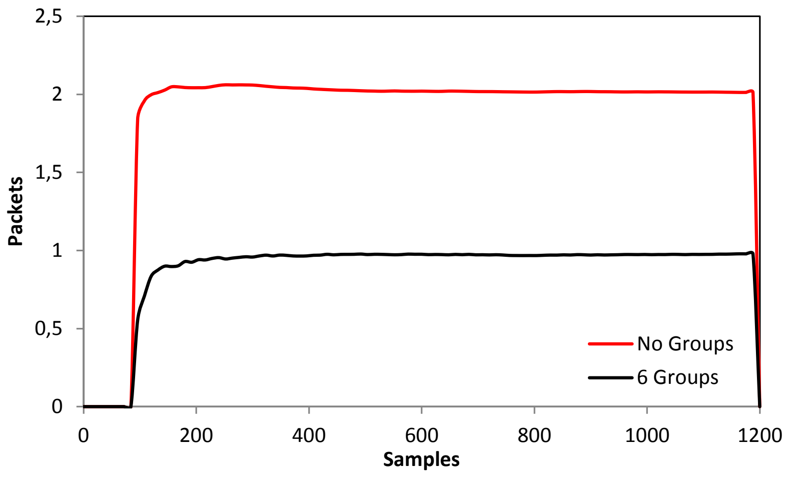


Figure 8-16: Retransmissions attempts.

8.6.3.4 *A forwarding sensor node in a network of 68 sensor nodes*

This subsection analyzes the performance of a forwarding node when it performs routing tasks. In **Figure 8-17**, we observe the traffic received by a random leaf sensor node when it routes the information sensed by other sensor nodes to the sink node. It shows that there are some peaks because a node can receive information from multiple nodes at the same time and our simulator does not differentiate between arrivals. We noticed that the data processed by the node does not exceed 90 bps. It is an appropriate bit rate for the selected type of sensor node and for the acoustic underwater communication.

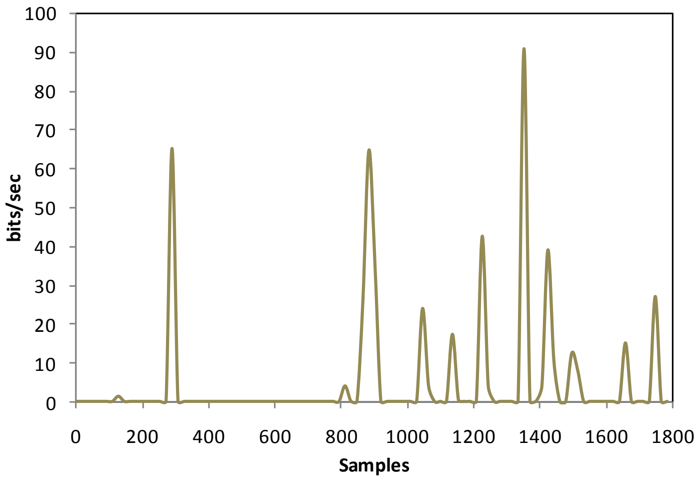


Figure 8-17: Traffic received in a random leaf sensor node.

8.6.3.5 <u>*A forwarding sensor node in a network of 200 sensor nodes*</u>

In this subsection we analyze the performance of a random leaf sensor node when the network has 200 sensor nodes organized in groups. **Figure 8-18** shows the traffic received by a random sensor node when another sensor node sends the sensed data to the sink node. We observe the same behavior than **Figure 8-18**. That is, the node receives data from multiple nodes. The main difference is that the average value is higher. The amount of data processed by the node does not exceed 225 bps. It is also an appropriate bit rate for acoustic underwater communications.

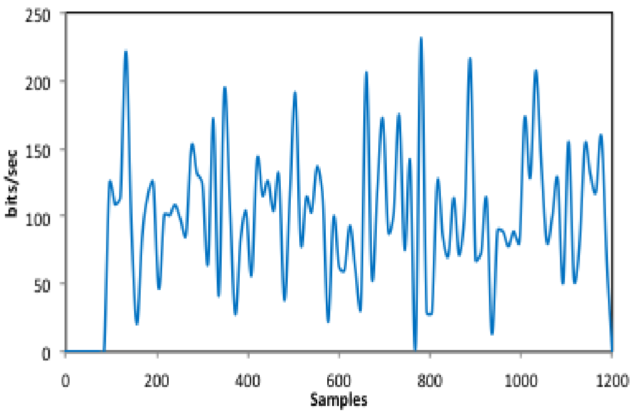


Figure 8-18: Traffic received in a random leaf sensor node.

8.7 Distributed sensor data fusion system

In this section we describe our automatic feeding system for each cage in marine fish farms. This system is based on an optimal decision through the fusion of the data obtained from the sensors distributed throughout the cage. First we describe the architecture of the proposed system. Then, we explain its algorithm operation. Finally, we will make a statistical analysis to analyze its performance.

8.7.1 System architecture and its operation

The architecture of our system is shown in **Figure 8-19**. In this figure we can see that the system consists of a set of distributed sensors that are connected to a management system that is responsible for making decisions. The system presented is composed of several sensing systems (temperature sensors, oxygen sensors, speed sensors, biomass distribution placement sensors and pellet detection sensor). Each sensor sends the collected data to the management system. The management system is in charge of choosing the appropriate action based on an algorithm (explained later) that depends on the sensed data. Each cage has one or more food dispensers. We designed four speeds for the food dispensers. The management system indicates which speed should be executed every time: regular, slow, very slow and stop. Because the system has various feeding rates, the system can dispense food according to the satisfaction of the fish in the cage. Now, it is not necessary to wait until there is surplus food at the bottom of the cage in order to stop the feeding process. The system is adaptive and incremental. When the fish are hungry, the system dispenses food regularly, and later, it will vary the speed until it stops the feeding process.

All sensors in the fish cage are connected to the management system. All sensors have a unique identifier (sID), which is sent jointly with each data to the management system. Thus, the system will identify each value with the source sensor. Moreover, the sensed data can be stored in a database. This data could be used to study the behavior of the fishes in different study cases and given conditions.

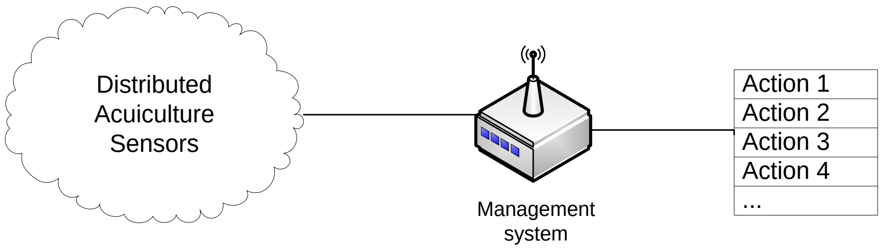


Figure 8-19: Architecture of our system.

The proposed system is based on simple operations (see **Figure 8-20**), where the sensed data is mixed to create a complex system. The proposed automatic feeding system consists of the following parts:

1. At the first level we have some physical parameters whose values are gathered from the fish cage. These physical parameters are measured from the medium.
2. At the second level, we have the sensors. They transform a magnitude to a value understandable to the system. In this system there are six groups of sensors: water column temperature sensors, oxygen column sensors, biomass distribution placement sensors, speed sensors, pellet detection sensors and outside sensors (presence of detection sensors and the water current speed sensors).
3. At the third level, the management system is placed. It is the core of our proposal. This module collects the data from the sensors and applies the proposed algorithm (explained later). In this system the data fusion process takes place. This process selects the output according to the data gathered by the sensors.
4. Finally, at the fourth level, there are four actions. These actions are used by the automatic feeding system.

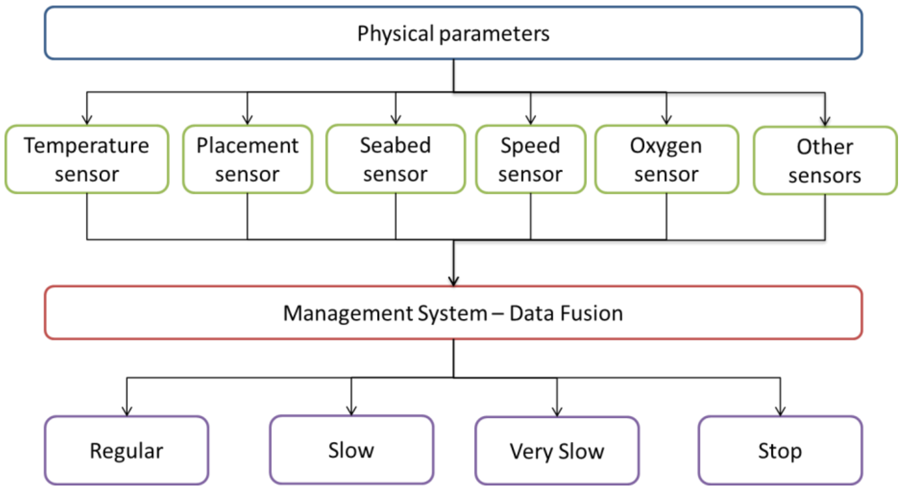


Figure 8-20: Block diagram of the proposed system.

The algorithm used by the management system is detailed in **Figure 8-21**. All operations performed by the management system are represented in this figure. The first task undertaken is to capture and store the data from the sensors. If the sensed data do not exceed certain thresholds, they will be stored in the database and the management system's output is 'STOP' (the feeding system will be stopped). The data coming from the sensors are tested to know whether they exceed a certain threshold. The thresholds located in the algorithm depend on the type of fish, the sea farm situation, the time of year, and so on. They should be introduced in the system previously and they are different for each fish species. Once the data from one sensor exceed a threshold, then other sensor data are analyzed to take the appropriate actions. According to the information gathered from other aquiculture works, the most related parameters are:

- The speed of the fishes and their distance from the water surface.
- The temperature of the water and the distance of the fishes from the water surface.
- The oxygen of water, the distance of the fishes from the water surface and the speed of the fishes.
- The speed of the fishes and the water current speed.
- The speed of the fishes and the presence of predators

All these parameters are linked and treated by a statistical process (this process will be seen in the next subsection). As shown in **Figure 8-21**, the data gathered from the sensors are always being monitored. When the sensed data gives that excess food is falling to the bottom, it clearly indicates that the fish are not eating; hence the feeding process must stop. This parameter also has a threshold but it is very low. The output of the statistical process with the data of the excess food will be the inputs for the selection process: this process indicates to the dispenser of food which is the appropriate speed depending on what is happening in the cage. Finally, the data from the sensors are stored in a database. This data can help to improve the thresholds and also be used to perform a study of the behavior of the fish in marine fish farms.

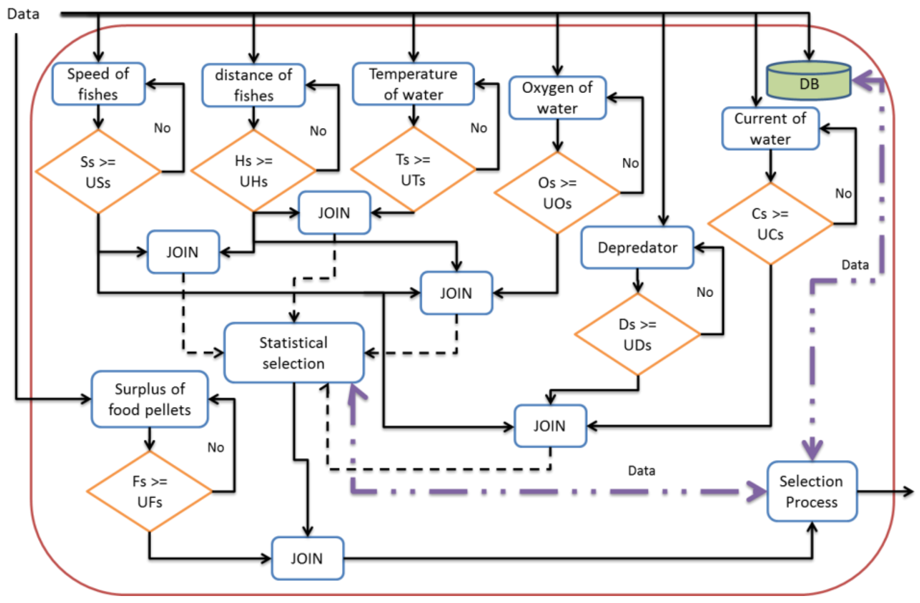


Figure 8-21: Detailed diagram of the algorithm.

8.7.2 Statistical analysis

The system should be able to change the action according to the behavior of the fish. But, our purpose is to go a step beyond; the system must be able to anticipate events, that is, it should be able to advise the food dispensers before the sensor detects excess food pellets. In order to achieve this goal, we must base our joining process on a statistical system. The system uses some preliminary data of the behavior of the species to feed according to the parameters used in our system. The parameters are season (E), water temperature (T), the distance of the fish (H), the speed of the fishes (S), the amount of oxygen in the water (O), water current (C), presence of a predator (D) and overeating (F).

We know the temperature water column for each season of the year for the placement of each species (these values are provided in a table later). There is a chance that the fishes have a speed tied to their distance to the surface. There is also a chance to increase the speed of the fish due to their distance to the surface and the amount of oxygen. Moreover, if there are low oxygen values in the water, the fish gets into a stress situation and it moves faster.

The speed of the fish also will be conditioned if there is a predator near the fish cage, and this speed will be calibrated because the water current will be taken into account. Finally, all of these probabilities are linked to the probability of having excess food in the bottom of the cage. This joint probability determines the type of fish feeding speed. **Figure 8-22** shows the explanation of the links between the parameters, for a better understanding of the joint probability.

The probability that indicates the feed rate (Px) is equal to the unconditional probability density when there is no overeating P(F) multiplied by the weighted λ_x sum (see Eq. 8-4). This weighted λ_x sum is the sum of all the aforementioned conditions (e.g. for the first case it will be λ_1 multiplied by the speed-conditional probability density of distance H multiplied by the prior probability P(H)).

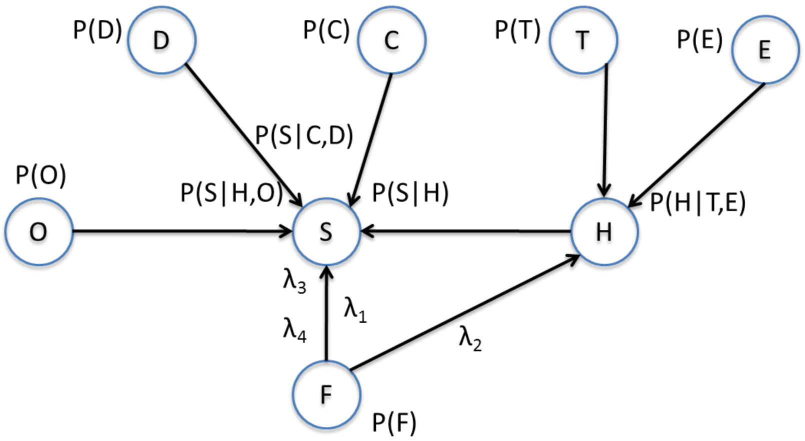


Figure 8-22: Detailed diagram of the algorithm.

$$P_x = P(F)\big[\lambda_1[P(S|H)P(H)] + \lambda_2[P(H|T,E)P(T,E)] + \lambda_3[P(S|H,O)P(H,0)] + \lambda_4[P(S|C,D)P(C,D)]\big] \quad \text{Eq. 8-4}$$

The sum of λ_x variables must be equal to 1, as is indicated in Eq. 8-5. λ_x variables let us give more priority to some elements than others.

$$\sum_{i=0}^{4} \lambda_i = 1 \quad \text{Eq. 8-5}$$

According to the output of the probability function shown in equation 1, the management system will execute an action. The values chosen to take the appropriate action are shown in **Table 8-3**.

Table 8-3: Actions taken according to output levels.

Output value	*Collection Method*
0 – 0.25	The feeding system should be stopped
0.25 – 0.5	Food dispenser at very slow speed
0.5 – 0.75	Food dispenser at slow speed
0.75 – 1	Food dispenser at regular speed (the fish is hungry)

8.7.3 Sensor data fusion system simulations

First we have gathered some information from the real world. **Table 8-4** and **Table 8-5** summarize the main fish species produced by marine fish farming. This information is taken from the Food and Agriculture Organization of the United Nations (FAO). A regular feed dispenser can feed from just a few grams of pellets every second up to 200–300 g/s. On-growing fish farm systems are delivered with the capacity of up to 3 kg/s. They can feed up to 45 tons each hour. The bigger on growing systems can dispense between 10 kg and 3 tons every hour [138].

In order to simulate our system, we have used Octave [139]. It is a high-level language created for numerical computations. It is mostly compatible with MATLAB. We suppose a regular marine fish cage with Sparus aurata and another marine fish cage with Dicentrarchus labrax in the Mediterranean zone. Both cages have a biomass of 2.8 kg inside. We have simulated our system during the summer and winter seasons. The fishes are fed during the summer season with a temperature range of 24.5–26.58ºC and during the winter season with a temperature range of 12–138ºC.

Table 8-4: Main fish species produced by marine fish farming (i).

Specie	***Distance from the surface (m)***	**Temperature range (ºC)**	***Sea***
Sparus aurata	1 – 30	14 - 26	Mediterranean
Dicentrarchus labrax	1 - 100	8 - 24	Mediterranean and Oriental Atlantic
Pagellus bogaraveo	150 - 300	12 - 21	Mediterranean and Oriental Atlantic
Argyrosomus regius	15 - 300	14 - 23	Mediterranean and Oriental Atlantic
Thunnus thynnus thynnus	0 - 985	18 - 24	Mediterranean and Atlantic
Salmo salar	0 - 210	2 - 9	Mediterranean and Atlantic
Gadus morhua	0 - 600	11 - 15	Mediterranean and Atlantic
Rachycentron canadum	0 - 1200	15 - 30.5	Worldwide in tropical and subtropical waters
Thunnus thynnus orientalis	1 - 200	17 - 23	North Pacific southeast Pacific
Thunnus albacares	1 - 100	15 - 31	Worldwide in tropical except Mediterranean
Thunnus obesus	0 - 250	13 - 29	Worldwide in tropical except Mediterranean
Thunnus maccoyii	50 - 2743	5 - 20	South Atlantic, South Indic and South Pacific
Diplodus puntazzo	0 - 60	14 - 29	Mediterranean
Umbrina cirrosa	0 - 100	10 - 30	Mediterranean
Pagrus major	10 - 200	10 - 22	Northeastern part of South China Sea northward to Japan (Philippines excluded)
Pagrus pagrus	10 - 80	12 - 18.5	Inter tropical

Table 8-5: Main fish species produced by marine fish farming (ii).

Specie	***Distance from the surface (m)***	**Temperature range (ºC)**	***Sea***
Sciaenops ocellatus	0 - 45	24 - 31	Western Atlantic
Anguilla anguilla	0 - 700	18 – 25	Mediterranean and Oriental Atlantic
Pagellus erythrinus	20 - 100	25 - 28	Mediterranean and Oriental Atlantic
Diplodus sargus	0 - 50	19 – 23	Mediterranean and Occidental Africa
Mugil cephalus	0 - 10	8 - 24	Inter tropical
Dentex dentex	15 - 50	14 – 24	Mediterranean
Larimichthys polyactis	0 - 120	3 - 12	Yellow and East China seas
Centroberyx affinis	10 - 450	16 - 24	Western Pacific
Chanos chanos	1 - 30	15 - 43	Indo-Pacific
Seriola quinqueradiata	100-1176	20 - 29	Northwest Pacific
Seriola dumerili	18 - 72	13 - 24	Inter tropical

Figure 8-23 shows the system performance with Sparus aurata for summer and winter seasons. In a regular procedure, the fishes are fed during 75 min. The feeding of the Sparus aurata is more stable in summer and they get to eat 3 tons of pellets each time (approximately 222 g/s), whereas the Sparusaurata feeding is much more variable with a mean value of 500 kg (approximately 185 g/s). At 65 min the system detects enough biomass moved to the bottom (although there are some fishes satisfied before), so, in our system, the dispenser speed changes its speed to slow. This behavior is also repeated at 72 min. Finally at 79 min our system stops the dispenser. We can see that our feeding process changes the feeding speed of the dispenser as the fishes are satisfied, which saves more food.

For Dicentrarchus labrax, system performance is in **Figure 8-24**. In this case Dicentrarchu slabrax has higher variations between summer and winter. They eat a mean value of 2.5 tons of pellets (approximately 556 g/s) in summer time. Both feeding processes are similar. In winter, the feeding of Dicentrarchus labrax is lower; they eat 600 kg each time (approximately 222 g/s). The system behavior is the same as the one explained before. At 65 min the system detects enough biomass moved to the bottom, so the dispenser speed changes its speed to slow. This behavior is also repeated at 72 min. Finally at 79 min our system stops the dispenser. We can see that our system begins to save food before the regular procedure stops.

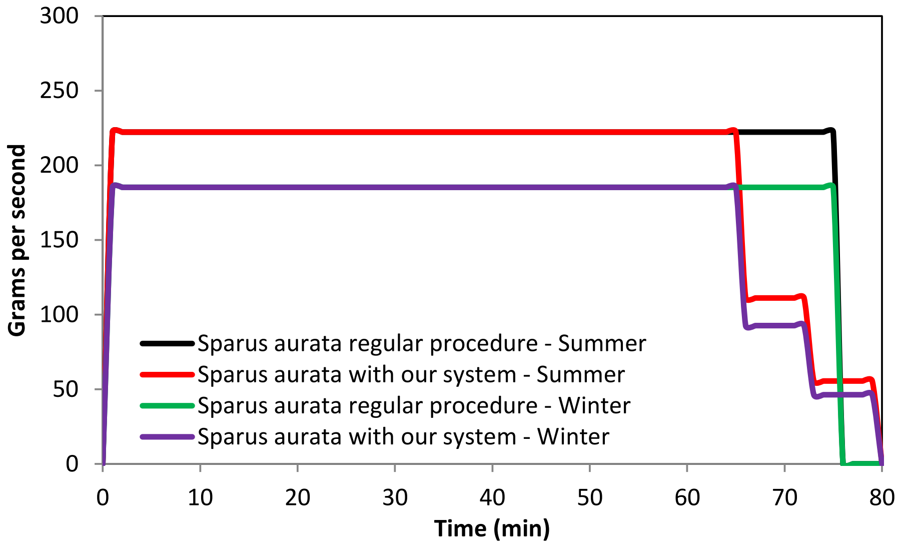


Figure 8-23: System performance with Sparus aurata.

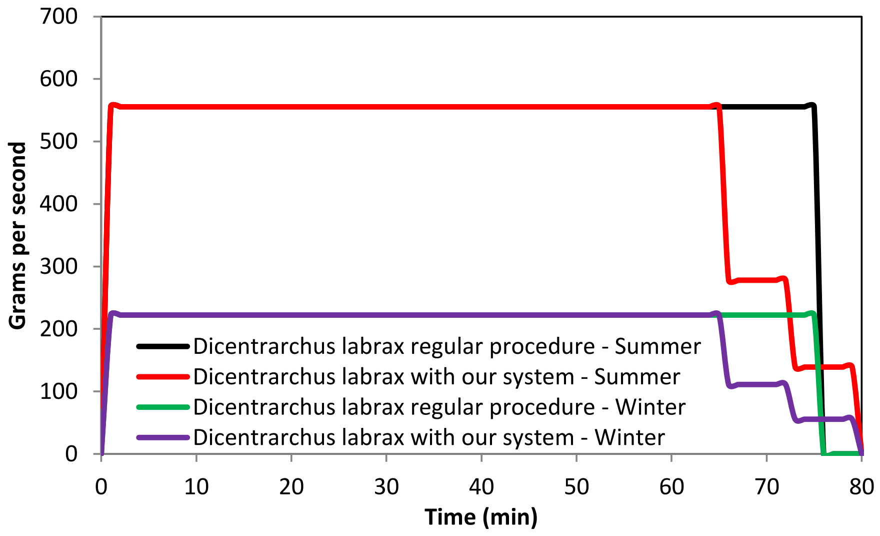


Figure 8-24: System performance with Dicentrarchus labrax.

Pellets come in many different sizes; from tiny ones (around 2 mm in diameter) to big ones (around 28 mm of diameter). Both Sparus aurata and Dicentrarchus labrax usually eat 6–6.5 mm pellets when they have a weight of 500 g. A regular pellet with this size has a mean weight of 0.5 g.

Figure 8-25shows the number of pellets that have fallen to the seabed for Sparus aurata during our automatic management feeding process compared to a regular process (using a table with time values or observing that there are no hungry fishes on the surface) for summer and winter time. We can observe that fewer numbers of pellets fell to the seabed in our proposal. Moreover, at the end of the feeding process, substantial food is saved. With a regular process we can see that we can detect until 160 pellets in the seabed (maximum value) while our proposal the maximum is around 10 pellets.

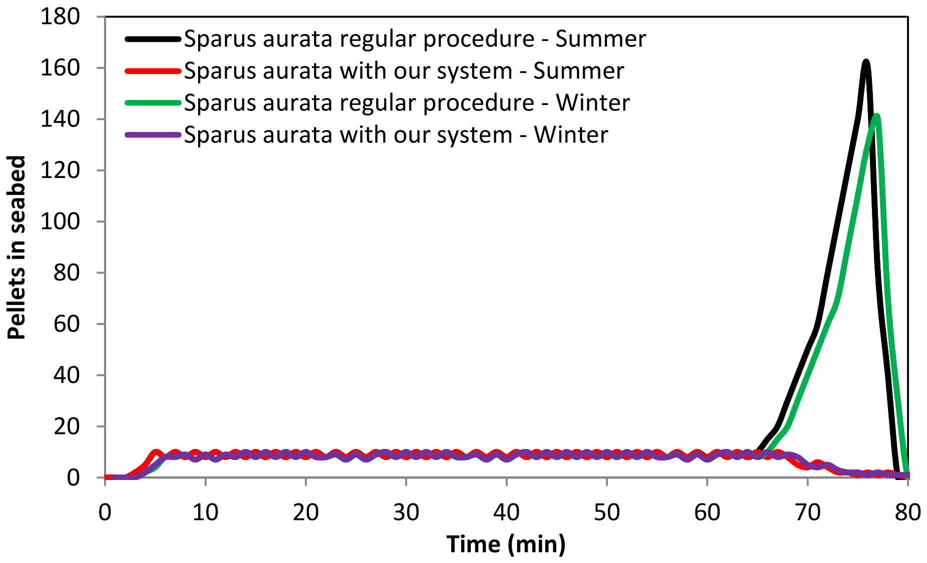


Figure 8-25: Pellets in the seabed with Sparus aurata.

Figure 8-26 shows the number of pellets which have fallen to the seabed for Dicentrarchus labrax during our automatic management feeding process compared to a regular process (using a table with time values or observing that there are no hungry fishes on the surface) for both seasons. We can observe that fewer pellets fell to the seabed in our proposal. With a regular process we can perceive that we can detect until 320 pellets in the seabed (maximum value) while our proposal the maximum is around 10 pellets. We can see that at the end of the feeding process the difference of the number of pellets in the seabed is substantially lower.

In **Figure 8-25** and **Figure 8-26** maximum values are related to fish feeding. When the fishes needs more food in a regular process there are more pellets in the seabed, but this effect does not happen when our proposal is running. It helps pellets' deposition to be independent of fish feeding.

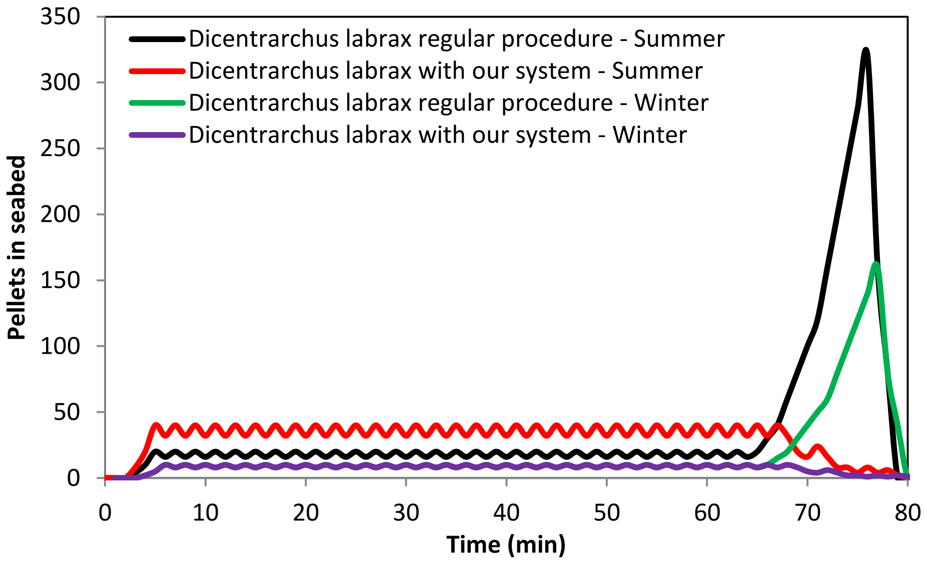


Figure 8-26: Pellets in the seabed with Dicentrarchus labrax.

8.8 Conclusion

As we have seen in this chapter, one of the main issues to take into account for the sustainability in marine fish farms is the economic losses and environmental impact produced by the fecal pellets and uneaten feeds that are spread under the cages. Parameters such as the depth under the cage, the mean current speed, the current direction, the settling speed, the position of each cage and the physic-chemical characterization of the water could be used to estimate the pollution level in an area. We have proposed a new system to control a fish farm. This system is composed of a group of sensor located in each cage and a group-based underwater wireless sensor network where sensors are mobile under a limited space, but fixed with a wire to the seabed, in order to enlarge the sensed area.

Our initial study was divided into two parts. In the first one, we have estimated analytically the sedimentation as a function of the distance for a sinking speed per sinking time equal to 10 and equal to 5. In both cases, we observed that the concentration of sediments is bigger in the center of each cage and this value decreases when the distance from the center of the cage increases. For this reason, we decreased the number of sensors in the peripheral areas. Following all our estimations we have distributed the sensors uniformly, in each sensing zone. The second part of this work was focused on the design of the group-based underwater wireless sensor network. The network topology is based on a fish farm composed by 6 cages (easily scalable). We define three types of nodes, regular sensor nodes (RS), border sensor nodes (BS) and sink sensor nodes (SS), with different roles in order to have an efficient communication between groups.

We have used two 6 groups-based topologies in order to simulate the performance of our proposal, one with 68 sensor nodes and another with 200 sensor nodes. In both cases the traffic load of the network did not exceeded 700 kbps. We also analyzed the traffic registered by a random intermediate sensor node in order to verify if it was able to perform its routing tasks. We can conclude, that the obtained values are appropriates for the selected type of sensor node and for acoustic underwater communication.

Compared to other proposals shown in the related work section, our system takes the measurements directly from the fish cage (not by simulation), which allows us to take an effective control on the feeding mechanism. Moreover, the number of people controlling the feeding process is reduced because it is controlled automatically. The proposed system is based on underwater sensors that can be brought on the market, and it can be complementary with other controlling systems (e.g. the use of subaquatic fixed cameras inside the fish cage). Due to different fish species have different behavior, thus give different parameter values. We have used sensed data fusion in order to avoid false positives.

Finally, work presented in this chapter has been published in the following references [36], [29], [28] and [30].

Chapter 9

CONCLUSION AND FUTURE WORK

9.1 Introduction

In this chapter we will present the main conclusions of this thesis and future work related to it.

As at the end of each chapter we have presented the summaries and conclusions about topics discussed in each of them, in this chapter we will present these findings with a global perspective.

We can see at the beginning of this thesis that WSNs are needed to improve the quality of people's life. They will be probably a basic piece of the future medical applications. These reasons make WSNs a main research area for many research groups. WSNs help the humans to control, examine and survey places that they are not able to do because it is very difficult to achieve the place or because humans are not able to measure by themselves.

In this thesis, we have proposed a creation of a group-based architecture, where nodes will have the same functionality within the network. Each group will have a head node, which defines the area in which nodes are within its group, but their functionality is the same as other nodes. Each node has a unique node identifier (*nodeID*). First group's node makes a group identifier (*groupID*). The main feature of this thesis is to create a group-based architecture to improve the efficiency of WSNs, regardless of the protocols that are being implemented in the lower layers. This architecture is located over the transport layer, so it does not depends on any routing protocol. As we could see in Chapter 2, every protocol based on groups or clusters uses its own routing protocol. We use groups instead of cluster because cluster-based networks are a subset of the group-based networks. Every cluster could be considered as a group. But a group-based network is capable of having any type of topology inside the group, not only clusters.

In order to carry out this architecture, firstly we had to check that group-based architectures could improve the behavior of WSNs. For this reason, we have simulated DSR, AODV and OLSR protocols with and without groups and the results show that group-based topologies give better performance for wireless ad-hoc networks. So, grouping nodes increases the productivity and the performance of the network with low overhead and low extra network traffic. Therefore, good scalability can be achieved in group-based networks.

When we saw that a group-based architecture could be able to improve the regular WSNs, we start to develop a new simple group-based protocol. This protocol should be able to create and to manage groups, but always taking into account that it respects the features of WSNs.

The implementation of this group-based protocol is the first step to use our proposal in a real wireless sensor network. For this reason we have implemented this protocol and we have tested it in a network simulator, which emulates the communication layers that are below than our proposal.

Viewing that this proposal was feasible, we thought in to include more features that could improve it. The first feature added was collaborating between groups. With this feature our groups can share data and they take decisions according to information sensed by nodes of other groups. Moreover, other necessary feature was security. Our network has to be able of share information in secure way, so we introduce security in our group-based proposal and finally we add synchronization. An essential feature in WSNs in order to improve the saving energy, with these techniques all sensor nodes are synchronized and they can in on-mode or standby-mode at the same time.

In the next subsection we show the main conclusions of this thesis following the order of chapters.

9.2 Conclusions and Contributions

As we have seen a group-based architecture provides some benefits for the whole network such as the content availability is increased because it could be replicated to other groups, it provides fault tolerance because other groups could carry out tasks from a failed group and it is very scalable because a new group could be added to the system easily. On the other hand, a group-based network can significantly decrease the communication cost between end-hosts by ensuring that a message reaches its destination with small overhead and highly efficient forwarding. Grouping nodes increases the productivity and the performance of the network with low overhead and low extra network traffic.

For these reasons we have defined a group-based architecture where links between groups could be established by physical proximity plus neighbor node capacity (λ). Its operation, maintenance and fault tolerance have been detailed. Messages designed to work properly have been shown. All simulations show its viability and how it could be designed to improve its performance. Finally, the designed group-based architecture has been compared with other similar architectures. In this comparison we have been able to see that our group-based proposal improves the previous architectures done by other authors.

Then, this group-based protocol has been implemented. Its classes and their relation have been explained. We have tested this implementation using a simulator and our results noted that our proposal works in a good way when in our topologies we have a node's activation process, which follows an exponential distribution. This feature provides an advantage to this architecture, as we say does not depend on the physical topology of the nodes or their activation process.

Besides, we include collaboration between groups. This idea is based on exchange information between groups, and then it could be used to change the direction of the alert propagation and the level of the alert in order to take the appropriate actions. This communication is based on a collaborative intergroup routing pool tree where the end nodes exchange information about the neighbor's groups. This exchanged data creates a collaborative routing pool tree where the root is the sink node and there is collaboration between the neighbor's groups. This intragroup communication improves the energy efficiency.

Then, we have added security in our group-based architecture. Our proposal has two security zones. On the one hand we have the intragroup security and, on the other hand, the intergroup security. Intergroup security of network is based on the symmetrical cryptography with a unique key for all the members of the network. Intragroup security is made by group keys. These keys are created through pseudorandom functions and they are modified by an update value every time that there is a change in the group. We have estimated the energy consumption given by each operation in this secure protocol for a group-based architecture and we have compared the energy consumption of our protocol when it is used in the elected sensor with other sensor devices. Results show us that this new feature needs more energy but if we compare this system with others, it could be carried out without any problem.

We have based our proposal on receiver-receiver protocol, but adapting the main idea to a group-based topology. Our proposal has been simulated for the same scenario changing μ and σ values of our time arrival distribution. All cases show that our proposal can resolve the offset time between nodes in any moment. But this offset time is more stable when as the number of samples is going up.

Finally, we have though proper to design a system where our group-based proposal or a modification of itself could work correctly. As our research institute is working in marine topics, we thought in a new system to control a fish farm. This system is composed of a group of sensor located in each cage and a group-based underwater wireless sensor network where sensors are mobile under a limited space, but fixed with a wire to the seabed, in order to enlarge the sensed area. The main of this work was focused on the design of the group-based underwater wireless sensor network. The network topology is based on a fish farm composed by 6 cages (easily scalable). We define three types of nodes, regular sensor nodes (RS), border sensor nodes (BS) and sink sensor nodes (SS), with different roles in order to have an efficient communication between groups.

We have used two 6 groups-based topologies in order to simulate the performance of our proposal, one with 68 sensor nodes and another with 200 sensor nodes. In both cases the traffic load of the network did not exceeded 700 kbps. We also analyzed the traffic registered by a random intermediate sensor node in order to verify if it was able to perform its routing tasks. We can conclude, that the obtained values are appropriates for the selected type of sensor node and for acoustic underwater communication.

Compared to other proposals shown in the related work section, our system takes the measurements directly from the fish cage (not by simulation), which allows us to take an effective control on the feeding mechanism. Moreover, the number of people controlling the feeding process is reduced because it is controlled automatically. The proposed system is based on underwater sensors that can be brought on the market, and it can be complementary with other controlling systems (e.g. the use of subaquatic fixed cameras inside the fish cage).

From this thesis we have done several scientific contributions (conference papers, book chapters and journal papers) as we have seen at the end of each chapter. The main papers of this thesis are [20], [19], [24], [27], [26], [14], [29] and [28].

But not everything has been perfect for the realization of this thesis. In our opinion, there are some disadvantages and flaws in this thesis. The main disadvantage of creating a group-based architecture for WSNs is the overhead introduced by a management protocol above the transport layer. Sensor nodes have to process this control traffic. If this traffic is very high, the benefits provided by the grouping feature may become negligible. Therefore, there is a commitment that has been shown in **Table 6-2**. In Chapter 8 we propose a situation where we think that a group-based wireless sensor network could work properly. Moreover, this implementation would provide a great benefit to businesses and sustainability of the marine environment. But this implementation we have not already finished. Currently, our research group is starting to work in this implementation on a fish farm.

9.3 Future Research Work

According to the main idea of this thesis, which is the division of nodes into groups, where each group is independent from the rest and all groups together make a large network a national R&D project was born. The main objective of this R&D project (TEC2011-27516) is to design and develop a new group-based wireless sensor network that uses the behavior of the environment (aquatic medium) in order to monitor it and make predictions. In order to achieve this goal we are applying self-learning algorithms, representation of the knowledge acquired from the environment and the information management of the wireless sensor network.

A future research work could be to design and development of a proper protocol, a collaborative data acquisition system, and appropriated decision algorithms. With a collaborative group-based network gives better performance to the group and to the whole system, thereby avoiding unnecessary message forwarding and additional overheads while saves energy. Grouping nodes also diminishes the average network delay while allows to scale the network considerably. In order to offer an optimized monitoring process, and in order to offer the best reply in particular environments, group-based collaborative systems are needed. They will simplify the monitoring needs while offering direct control.

Another aspect could be to add intelligence to the group-based WSN. We will use a cognitive group-based collaborative wireless sensor network where every node will gather measures from two sides. The first one could be the direct sensing method, which obtains the variables sensed from the physical medium, that is, the information measured physically. The second one could be the indirect sensing method, where the measures would be obtained from parameters of the wireless network (Received Signal Strength, lost frames, disconnections, etc.) in order to extrapolate some conclusions useful to have a complete picture of what is happening in the aquatic medium and obtain the right prediction.

In order to achieve this future work, we should study how some data network parameters are affected by what is happening in the environment, thus, we will study which are the most useful parameters for the environmental monitoring and their relationship with what is happening in the environment. Moreover, when a node has a new event, it warns the alert, jointly with the variables and parameters measured, to its group, which propagates the information to its neighboring groups if needed (based on some decisions previously established). According this cooperation, the sensor network will be efficient and the sensors will have a longer lifetime. With these new features, we could add mechanisms that allow using the measurements obtained from the parameters of the cognitive group-based collaborative sensor network to improve the performance of the network.

Another important aspect to take into account possible future work would be to improve the security in group-based WSNs. In this thesis, we only take into account the privacy of the data transmitted. But, if we want to have a secure network, we should take into account that a network must accomplish the following features: confidentiality, integrity, authenticity, and nonrepudiation. Improving the other security features would be very important in order to have a solution when there are some nodes act maliciously or are compromised.

Following the idea presented in Chapter 8, the use of a sensor network to create an automatic system for the management of marine fish farms could be a very important aspect for several companies who currently own fish farms. Due mainly for two reasons: a) control of the seabed pollution near fish farms and b) the amount of money that they could save over the years. The best future work related to this feature. It would be focused on the implementation and operation of the proposed system. Nowadays, there is not any similar proposal, so it could be a good product for the marine industry.

Finally, monitoring and detection of fires may be another application field where the proposed architecture could work. In this application the network should include the collaborative feature presented in Chapter 6. The main idea would be to distribute sensor nodes in strategic locations in the monitored area. After that, the network would create groups according to its requirements and its groups start to collaborate between them. The idea would be to allow different levels of alert depending on the degree of danger of the sensed parameter and the position of the neighboring groups. In this type of WSNs when a sensor detects a new event, the alert would be sent to its group and it is distributed to an appropriate neighboring groups based on the information shared between sensors.

BIBLIOGRAPHY

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. A. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine,* vol. 40, pp. 102-114, 2002.

[2] M. Frodigh, P. Johansson y P. Larsson, «Wireless ad hoc networking. The art of networking without a network,» *Ericsson Review,* vol. 4, pp. 248-263, 2000.

[3] S. Kumar, V. S. Raghavan and J. Deng, "Medium access control protocols for ad hoc wireless networks: A survey," *Ad Hoc Networks,* vol. 4, no. 3, pp. 326-358, 2006.

[4] M. Abolhasan, T. Wysocki and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks,* vol. 2, no. 1, pp. 1-22, 2004.

[5] J. A. Garcia-Macias y J. Gomez, «MANET versus WSN,» de *Sensor Networks and Configuration*, Berlin, Springer Berlin Heidelberg, 2007, pp. 369-388.

[6] T. Fencl, P. Burget and J. Bilek, "Network topology design," *Control Engineering Practice,* vol. 12, no. 11, pp. 1287-1296, 2011.

[7] J. K. Kurose and K. W. Ross, Computer Networking: A Top-Down Approach, 6th ed., Canada: Pearson Education, 2012.

[8] J. Lloret, Arquitectura de interconexión de redes P2P parcialmente centralizadas, 1st ed., Valencia: Universidad Politecnica de Valencia, 2006.

[9] J. Lloret, M. Garcia, D. Bri and J. R. Diaz, "A Cluster-Based Architecture to Structure the Topology of Parallel Wireless Sensor Networks," *Sensors,* vol. 9, no. 12, pp. 10513-10544, 2009.

[10] J. V. Capella, Redes inalámbricas de sensores: Una nueva arquitectura eficiente y robusta basada en jerarquía dinámica de grupos, Valencia: Universidad Politécnica de Valencia, 2010.

[11] J. Lloret, M. Garcia, F. Boronat and J. Tomas, "Group-based self-organization grid architecture," in *2nd International Conference on Advances in Grid and Pervasive Computing*, Paris, France, 2007.

[12] J. Lloret, M. Garcia and J. Tomas, "A Group-Based Architecture for Wireless Sensor Networks," in *3rd Int. Conference on Networking and Services. ICNS 2007*, Athens, Greece, 2007.

[13] M. Garcia, D. Bri, F. Boronat and J. Lloret, "A new neighbour selection strategy for group-based wireless sensor networks," in *4th int. conf. on networking and services, ICNS 2008*, Gosier, Guadeloupe, 2008.

[14] J. Lloret, M. Garcia and J. Tomas, "Improving Mobile and Ad-hoc Networks performance using Group-Based Topologies," in *Wireless Sensor and Actor Networks II*, vol. 264, Boston, Springer, 2008, pp. 209-220.

[15] J. Lloret, M. Garcia, F. Boronat and J. Tomas, "MANET protocols performance in group-based networks," in *IFIP International Federation for Information Processing*, vol. 284, Boston, Springer, 2008, pp. 161-172.

[16] M. Garcia, "Estudio del rendimiento de arquitecturas basadas en grupos para WAHSN," Universitat Politècnica de Valencia, Valencia, 2008.

[17] M. Garcia, H. Coll, D. Bri and J. Lloret, "Using MANET protocols in Wireless Sensor and Actor Networks," in *The Second International Conference on Sensor Technologies and Applications (SENSORCOMM 2008)*, Cap Esterel, France, 2008.

[18] J. Lloret, M. Garcia, F. Boronat and J. Tomas, "A Group-Based Protocol for Large Wireless AD-HOC and Sensor Networks," in *IEEE Network Operations and Management Symposium Workshops. NOMS 2008*, Salvador - Bahia, Brasil, 2008.

[19] J. Lloret, M. Garcia, F. Boronat and J. Tomas, "A Planar Group-Based Architecture to Scale Ad-Hoc and Sensor Networks," *Journal of Networks,* vol. 4, no. 6, pp. 473-486, 2009.

[20] J. Lloret, M. Garcia, J. Tomas and F. Boronat, "GBP-WAHSN: a group-based protocol for large wireless ad hoc and sensor networks," *Journal of Computer Science and Technology,* vol. 23, no. 3, pp. 461-480, 2008.

[21] M. Garcia and J. Lloret, "A Cooperative Group-Based Sensor Network for Environmental Monitoring," *Lecture Notes in Computer Science,* vol. 5738/2009, pp. 276-279, 2009.

[22] M. Garcia, S. Sendra, J. Lloret and R. Lacuesta, "Saving energy with cooperative group-based wireless sensor networks," in *7th International Conference Cooperative Design, Visualization, and Engineering*, Mallorca, Spain, 2010.

[23] M. Garcia, J. Lloret, S. Sendra and J. P. C. Rodrigues, "Taking Cooperative Decisions in Group-Based Wireless Sensor Networks," in *8th International Conference on Cooperative Design, Visualization and Engineering*, Hong Kong, China, 2011.

[24] M. Garcia, S. Sendra, J. Lloret and A. Canovas, "Saving energy and improving communications using cooperative group-based Wireless Sensor Networks," *Telecommunication Systems,* pp. 1-14, 2011.

[25] M. Garcia, D. Bri, J. Lloret and P. Lorenz, "Collaborating Using Intergroup Communications in Group-Based Wireless Sensor Networks: Another Way for Saving Energy," in *9th International Conference Cooperative Design, Visualization, and Engineering*, Osaka, Japan, 2012.

[26] M. Garcia, J. Lloret, S. Sendra and R. Laquesta, "Secure Communications in Group-based Wireless Sensor Networks," *International Journal of Communication Networks and Information Security (IJCNIS),* vol. 2, no. 1, pp. 8-14, 2010.

[27] M. Garcia, D. Bri, J. Lloret and P. Lorenz, "A Secure Intragroup Time Synchronization technique to improve the security and performance of Group-based Wireless Sensor Networks," in *Wireless Networks and Security*, In press, Springer, 2012.

[28] J. Lloret, S. Sendra, M. Garcia and G. Lloret, "Group-based underwater wireless sensor network for marine fish farms," in *IEEE Globecom Workshops 2011*, Houston, TX, USA, 2011.

[29] M. Garcia, S. Sendra, J. Lloret and G. Lloret, "Monitoring and control sensor system for fish feeding in marine fish farms," *IET Communications,* vol. 5, no. 12, pp. 1682-1690, 2011.

[30] J. Lloret, M. Garcia, S. Sendra and G. Lloret, "An Underwater Wireless Group-Based Sensor Network for Marine Fish Farms Sustainability Monitoring," *Telecommunication Systems,* vol. In press, 2014.

[31] J. Lloret, S. Sendra, H. Coll and M. Garcia, "Saving energy in wireless local area sensor networks," *The computer journal,* vol. 53, no. 10, pp. 1628-1673, 2009.

[32] M. Garcia, D. Bri, S. Sendra and J. Lloret, "Practical Deployments of Wireless Sensor Networks: a Survey," *International Journal on Advances in Networks and Services,* vol. 3, no. 1&2, pp. 170-185, 2010.

[33] S. Sendra, J. Lloret, M. Garcia and J. F. Toledo, "Power Saving and Energy Optimization Techniques for Wireless Sensor Neworks," *Journal of Communications,* vol. 6, no. 6, pp. 439-459, 2011.

[34] J. Lloret, M. Garcia, H. Coll and M. Edo, "Wireless Sensor Networks and Systems," in *Wireless Technologies: Concepts, Methodologies, Tools and Applications*, Pennsylvania, IGI Global, 2012, pp. 33-45.

[35] R. Lacuesta, M. Garcia, J. Lloret and G. Palacios, "Study and Performance of Ad Hoc Routing Protocols," in *Mobile Ad hoc Networks: Current Status and Future Trends*, NY, USA, CRC Press, Taylor and Francis, 2011, pp. 71-101.

[36] M. Garcia, S. Sendra, M. Atenas and J. Lloret, "Underwater Wireless Ad Hoc Networks: A Survey," in *Mobile Ad Hoc Networks. Current Status and Future Trends*, Boca Raton, FL: CRC Press, 2011, pp. 379-412.

[37] D. Bri, M. Garcia, J. Lloret and P. Dini, "Real Deployments of Wireless Sensor," in *Third International Conference on Sensor Technologies and Applications*, Athens, Greece, 2009.

[38] K. Woolston and S. Albin, "The design of centralized networks with reliability and availability constraints.," *Computers and Operations Research,* vol. 15, no. 3, pp. 207-217, 1988.

[39] L. S. Huang, H. L. Xu, Y. Wang, J. M. Wu and H. Li, "Coverage and exposure paths in wireless sensor networks," *Journal of Computer Science and Technology,* vol. 21, no. 4, pp. 490-495, 2006.

[40] A. Ganz, C. M. Krishna, D. Tang and Z. J. Haas, "On optimal design of multitier wireless cellular systems," *Communications Magazine,* vol. 35, no. 2, pp. 88-93, 1997.

[41] A. Wierzbicki, R. Strzelecki, D. Swierczewski and M. Znojek, "Rhubarb: A tool for developing scalable and secure peer-to-peer applications," in *2nd IEEE Int. Conference on Peer-to-Peer Computing*, LinkÄoping, Sweden, 2002.

[42] Z. Xiang, Q. Zhang, W. Zhu, Z. Zhang and Y. Zhang, "Peer-to-peer based multimedia distribution service," *IEEE Transactions on Multimedia,* vol. 6, no. 2, pp. 343-355, 2004.

[43] L. Hongjun, L. Luo and Z. Zhifeng, "A structured hierarchical P2P model based on a rigorous binary tree code algorithm," *Future Generation Computer Systems,* vol. 23, no. 2, pp. 201-208, 2007.

[44] B. Thallner and H. Moser, "Topology control for fault-tolerant communication in highly dynamic wireless networks," in *3rd International Workshop on Intelligent Solutions in Embedded Systems*, Hamburg, Germany, 2005.

[45] J. Y. Yu and P. H. Chong, "A survey of clustering schemes for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials,* vol. 7, no. 1, pp. 32-48, 2005.

[46] B. Shen, S. Zhang and Y. Zhong, "Cluser-based routing protocol for wireless sensor networks," *Journal of software,* vol. 17, no. 7, pp. 1588-1600, 2006.

[47] W. Heinzelman, A. Chandrakasan and H. Balakrishman, "Energy-efficient communication protocol for wireless sensor," in *Proceeding of the Hawaii International Conference System Sciences*, Hawaii, USA, 2000.

[48] S. Lindsey y C. S. Raghavendra, «PEGASIS: power efficient gathering in sensor information systems,» de *Proceedings of the IEEE Aerospace Conference*, Montana, USA, 2002.

[49] A. Manjeshwar and D. P. Agrawal, "TEEN: a protocol for enhanced efficiency in wireless sensor networks," in *Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, San Francisco, CA, USA, 2001.

[50] A. Manjeshwar y D. P. Agrawal, «APTEEN: a hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks,» de *Proceedings of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, Ft. Lauderdale, FL, USA, 2002.

[51] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed energy conservation for ad hoc routing," in *7th Annual international conference on mobile computing and networking*, Rome, Italy, 2001.

[52] Z. J. Haas, M. R. Pearlman and P. Samar, "The zone routing protocol (ZRP) for ad hoc networks," IETF Internet Draft, Fremont, CA, USA, 2002.

[53] M. Musolesi and C. Mascolo, "A community based mobility model for ad hoc network research," in *2nd International Workshop on Multi-Hop Ad Hoc Networks: From Theory to Reality*, Florence, Italy, 2006.

[54] P. F. Tsuchiya, "The landmark hierarchy: A new hierarchy for routing in very large networks," *Computer Communication Review,* vol. 18, no. 4, pp. 35-42, 1988.

[55] T. Rekhter and T. Li, "A border gateway protocol 4 (BGP-4). RFC 1771," IETF, Fremont, CA, USA, 1995.

[56] G. Pei, M. Gerla and X. Hong, "LANMAR: Landmark routing for large scale wireless ad hoc networks with group mobility," in *6th Annual Int. Conference on Mobile Computing and Networking*, Boston, MA, USA, 2000.

[57] F. Garcia, J. Solano and I. Stojmenovic, "Connectivity Based k-Hop Clustering in Wireless Networks," *Telecommunication Systems,* vol. 22, no. 1-4, pp. 205-220, 2003.

[58] J. Lloret, C. Palau, F. Boronat and J. Tomas, "Improving networks using group-based topologies," *Computer Communications,* vol. 31, no. 4, pp. 3438-3450, 2008.

[59] S. S. Meiyappan, G. Frederiks and S. Hahn, "Dynamic power save techniques for next generation WLAN systems," in *Proceedings of the 38th southeastern symposium on system theory*, Cookeville, TN, USA, 2006.

[60] V. Raghunathan, C. Schurgers, S. Park and M. Srivastava, "Energy aware wireless microsensor networks," *IEEE Signal Processing Magazine,* vol. 19, no. 2, pp. 40-50, 2002.

[61] R. Min, M. Bhardwaj, S. H. Cho, E. Shih, A. Sinha, A. Wang and A. Chandrakasan, "Low power wireless sensor networks," in *Proceedings of international conference on VLSI design*, Bangalore, India, 2001.

[62] A. Salhieh, J. Weinmann, M. Kochha and L. Schwiebert, "Power efficient topologies for wireless sensor networks," in *Proceedings of the IEEE international conference on parallel processing*, Valencia, Spain, 2001.

[63] Q. Gao, K. Blow, D. Holding, I. Marshall and X. Peng, "Radio range adjustment for energy efficient wireless sensor networks," *Journal of Ad Hoc Networks,* vol. 4, no. 1, pp. 75-82, 2004.

[64] M. Younis, M. Youssef and K. Arisha, "Energy-aware routing in cluster-based sensor networks," in *Proceedings of the 10th IEEE international symposium on modeling, analysis, and simulation*, Washington, USA, 2002.

[65] M. Ye, C. Li, G. Chen and J. Wu, "An Energy Efficient Clustering Scheme in Wireless Sensor Networks," *Ad Hoc & Sensor Wireless Networks,* vol. 3, no. 2-3, pp. 99-119, 2007.

[66] F. Hu and N. K. Sharma, "Security considerations in ad hoc sensor networks," *Ad Hoc Networks,* vol. 3, no. 1, pp. 69-89, 2005.

[67] S. Slijepcevic, M. Potkonjak, V. Tsiatsis, S. Zimbeck and M. B. Srivastava, "On Communication Security in Wireless Ad-Hoc Sensor Networks," in *Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Pittsburgh, PA, USA, 2002.

[68] S. A. Camtepe and B. Yener, "Key distribution mechanisms for wireless sensor networks: a survey," Rensselaer Polytechnic Institute, Computer Science Department, NY, USA, 2005.

[69] D. Liu, P. Ning and W. Du, "Group-based key pre-distribution in wireless sensor networks," in *Proceedings of the 4th ACM workshop on Wireless security (WiSe '05)*, New York City, NY, USA, 2005.

[70] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan and C. Zhang, "Secure group communications for wireless networks," in *Military Communications Conference, 2001. MILCOM 2001.*, McLean, VA, 2001.

[71] V. Varadharajan, R. Shankaran and M. Hitchens, "Security for cluster based ad hoc networks," *Computer Communications,* vol. 27, no. 5, pp. 488-501, 2004.

[72] D. Mills, "Internet time synchronization: The network time protocol," *IEEE Trans. Communications,* vol. 39, no. 10, pp. 1482-1493, 1991.

[73] E. D. Kaplan, Understanding GPS: principles and applications, Boston: Artech House, 1996.

[74] B. M. Sadler and A. Swami, "Synchronization in Sensor Networks: an Overview," in *Military Communications Conference. MILCOM 2006*, Washington DC, 2006.

[75] J. Elson and D. Estrin, "Time Synchronization for Wireless Sensor Networks," in *15th International Parallel & Distributed Processing Symposium*, San Francisco, California, USA, 2001.

[76] J. Elson, L. Girod and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," in *5th ACM SIGOPS Operating Syst. Review*, Boston, MA, 2002.

[77] Y. Yang and Y. Sun, "Securing Time-Synchronization Protocols in Sensor Networks: Attack Detection and Self-Healing," in *IEEE Global Telecommunications Conference. GLOBECOM 2008*, New Orleans, LA, USA, 2008.

[78] S. Ganeriwal, S. Capkun, C. Han and M. B. Srivastava, "Secure time synchronization service for sensor networks," in *4th ACM workshop on Wireless Security (WiSe '05)*, New York, NY, USA, 2005.

[79] H. Li, Y. Zheng, M. Wen and K. Chen, "A Secure Time Synchronization Protocol for Sensor Network," *Lecture Notes in Computer Science,* vol. 4819/2007, pp. 515-526, 2007.

[80] J. Lloret, M. Garcia, D. Bri and S. Sendra, "A Wireless Sensor Network Deployment for Rural and Forest Fire Detection and Verification," *Sensors,* vol. 9, no. 11, pp. 8722-8747, 2009.

[81] J. Lloret, J. Tomas, M. Garcia and A. Canovas, "A hybrid stochastic approach for self-location of wireless sensors in indoor environments," *Sensors,* vol. 9, no. 5, pp. 3695-3712, 2009.

[82] N. Vlajic and D. Xia, "Wireless sensor networks: to cluster or not to cluster?," in *International Symposium on a World of Wireless, Mobile and Multimedia Networks, 2006. WoWMoM 2006.*, Buffalo, NY, USA, 2006.

[83] OPNET Technologies, Inc., " OPNET Modeler® Wireless Suite," 1 January 2012. [Online]. Available: http://www.opnet.com/solutions/network_rd/modeler_wireless.html. [Accessed 15 August 2012].

[84] M. Jiang, J. Li and Y. C. Tay, "Cluster Based Routing Protocol (CBRP)," IETF -

Internet draft, 1999.

[85] C. C. Chiang, H. K. Wu, W. Liu and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," in *IEEE SICON*, Singapore, 1997.

[86] J. Sá Silva, T. Camilo, P. Pinto, R. Ruivo, A. Rodrigues, F. Gaudencio and F. Boavida, "Multicast and IP Multicast Support in Wireless Sensor Networks," *Journal of Networks,* vol. 3, no. 3, pp. 19-26, 2008.

[87] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications,* vol. 11, no. 6, pp. 6-28, 2004.

[88] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol," IETF - RFC 3626, 2003.

[89] R. Cohen and S. Havlin, "Scale-Free Networks Are Ultrasmall," *Physical Review Letters,* vol. 90, no. 5, pp. 1-4, 2003.

[90] R. Cohen, K. Erez, D. ben-Avraham and S. Havlin, "Resilience of the Internet to Random Breakdowns," *Physical Review Letters,* vol. 85, no. 21, pp. 1-4, 2000.

[91] G. Hermann, "Mathematical investigations in network properties," in *IEEE Intelligent Engineering Systems*, Cruising on Mediterranean Sea, 2005.

[92] P. Van Roy and S. Haridi, Concepts, Techniques, and Models of Computer Programming, Cambridge, MA: The MIT Press, 2004.

[93] R. Bosman, J. Lukkien and R. Verhoeven, "An Integral Approach to Programming Sensor Networks," in *6th IEEE Consumer Communications and Networking Conference*, Las Vegas, Nevada, USA, 2009.

[94] R. Gummadi, O. Gnawali and R. Govindan, "Macro-programming Wireless Sensor Networks Using Kairos," *Lecture Notes in Computer Science,* vol. 3560, pp. 126-140, 2005.

[95] Wikipedia, "List of wireless sensor nodes," Wikipedia, 22 October 2012. [Online]. Available: http://en.wikipedia.org/wiki/List_of_wireless_sensor_nodes. [Accessed 12 December 2012].

[96] OMNeT++, «OMNeT++ Network Simulation Framework,» OMNeT++ Community, 14 11 2012. [En línea]. Available: http://www.omnetpp.org/. [Último acceso: 14 11 2012].

[97] A. Varga, "OMNeT++ User Guide," 2011. [Online]. Available: http://www.omnetpp.org/doc/omnetpp/UserGuide.pdf. [Accessed 14 11 2012].

[98] A. Varga, "OMNeT++ User Manual," 2011. [Online]. Available:

http://www.omnetpp.org/doc/omnetpp/Manual.pdf. [Accessed 14 11 2012].

[99] N. Vlajic and D. Xia, "Wireless sensor networks: to cluster or not to cluster?," in *International symposium on a world of wireless, mobile and multimedia networks*, Buffalo, NY, 2006.

[100] I. Stojmenovic, "Position based routing in ad hoc networks," *IEEE Communications Magazine,* vol. 40, no. 7, pp. 128-134, 2002.

[101] W. B. Heinzelman, A. P. Chandrakasan and H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks," *IEEE Transactions on Wireless Communications,* vol. 1, no. 4, pp. 660-670, 2002.

[102] M. Bhardwaj, T. Garnett and A. P. Chandrakasan, "Upper bounds on the lifetime of sensor networks," in *The IEEE International Conference on Communications*, Helsinki, 2001.

[103] A. Gibbons, Algorithmic graph theory, Cambridge: Cambridge University Press, 1985.

[104] Mathworks, Inc., "MATLAB," Mathworks, Inc., 29 October 2012. [Online]. Available: http://www.mathworks.com/products/matlab/.

[105] J. M. Bohli, C. Sorge and D. Westhoff, "Initial observations on economics, pricing,and penetration of the internet of things market," *SIGCOMM Comput. Commun. Rev.,* vol. 39, no. 2, pp. 50-55, 2009.

[106] A. Perring, J. Stankovic and D. Wagner, "Security in wireless sensor networks," *ACM Communications,* vol. 47, no. 6, pp. 53-57, 2004.

[107] Y. C. Wu, Q. Chaudhari and E. Serpedin, "Clock Synchronization of Wireless Sensor Networks," *IEEE Signal Processing Magazine,* vol. 28, no. 1, pp. 124-138, 2011.

[108] F. Boronat, J. Lloret and M. Garcia, "Multimedia group and inter-stream synchronization techniques: A comparative study," *Information Systems,* vol. 34, no. 1, pp. 108-131, 2009.

[109] M. Matsui, "New Block Encryption Algorithm MISTY," in *4th International Workshop In Fast Software Encryption*, Haifa, Israel, 1997.

[110] H. Ohta and M. Matsui, "RFC 2994: A Description of the MISTY1 Encryption Algorithm," IETF Network, Fremont, CA, USA, 2000.

[111] Y. W. Law, J. Doumen and P. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks," *ACM Trans. Sensor Networks,* vol. 2, no. 1, pp. 65-93, 2006.

[112] G. de Meulenaer, F. Gosset, F. Standaert and O. Pereira, "On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks," in *IEEE international Conference on Wireless & Mobile Computing, Networking & Communication. WIMOB 2008*, Washington, DC, USA, 2008.

[113] A. Ageev, "Time Synchronization and Energy Efficiency in Wireless Sensor Networks," 2010. [Online]. Available: http://eprints-phd.biblio.unitn.it/260/. [Accessed 29 11 2012].

[114] M. V. Goulão, C. A. Andrade, N. M. Gouveia, J. R. Gomes, V. M. Timóteo and F. Soares, "Evaluación de pérdidas de piensos en una piscifactoría en mar abierto y su uso en modelos del crecimiento de peces de cultivo y de la ración diaria," AquaTIC, 2001. [Online]. Available: http://www.revistaaquatic.com/aquatic/art.asp?t=h&c=113. [Accessed 15 11 2012].

[115] B. Basurco y G. Larrazabal, «Marine Fish Farming in Spain,» *Cahiers Options Méditerranéennes,* vol. 30, pp. 45-56, 2000.

[116] S. Porrello, P. Tomassetti, L. Manzueto, M. G. Finoia, E. Persia, I. Mercatali and P. Stipa, "The influence of marine cages on the sediment chemistry in the Western Mediterranean Sea," *Aquaculture,* vol. 249, p. 145–158, 2005.

[117] E. Mantzavrakos, M. Kornaros, G. Lyberatos and P. Kaspiris, "Impacts of a marine fish farm in Argolikos Gulf (Greece) on the water column and the sediment," *Desalination,* vol. 210, pp. 110-124, 2007.

[118] J. Heidemann, W. Ye, . J. Wills, A. Syed and Y. Li, "Research challenges and applications for underwater sensor networking," in *IEEE Wireless Communications and Networking Conference*, Las Vegas, Nevada, USA, 2006.

[119] C. J. Cromey, T. D. Thomas and K. D. Black, "DEPOMOD – Modeling the deposition and biological effects of waste solids from marinecage farms," *Aquaculture,* vol. 214, pp. 211-239, 2002.

[120] A. Stigebrandt, J. Aure , A. Ervik and P. Kupka Han, "Regulating the local environmental impact of intensive marine fish farming: III. A model for estimation of the holding capacity in the Modelling-Ongrowing fish farm-Monitoring system," *Aquaculture,* vol. 234, pp. 239-261, 2004.

[121] S. Han, Y. Kang, K. Park and M. Jang, "Design of Environment Monitoring System for Aquaculture Farms," in *Frontiers in the Convergence of Bioscience and Information Technologies*, Jeju Island, Korea, 2007.

[122] A. Pal, "Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges," *Network Protocols and Algorithms,* vol. 2, no. 1, pp. 45-73, 2010.

[123] R. Mulligan and H. M. Ammari, "Coverage in Wireless Sensor Networks: A Survey," *Network Protocols and Algorithms,* vol. 2, no. 2, pp. 27-53, 2010.

[124] I. Karakassis, M. Tsapakis, E. Hatziyanni, K. N. Papadopoulou and W. Plaiti, "Impact of cage farming of fish on the seabed in three Mediterranean coastal areas," *ICES Journal of Marine Science,* vol. 57, pp. 1462-1471, 2000.

[125] M. Jover, S. Martínez, A. Tomás and I. Pérez, Diseño y gestión de granjas acuícolas, Valencia, Spain: Universitat Politécnica de València, 2005.

[126] FAO, Good practices for the feed industry. Implementing the Codex Alimentarius Code of Practice on Good Animal Feeding, United Nations: FAO Animal Production and Health Manuals, 2010.

[127] G. Sarà, D. Scilipoti, A. Mazzola and A. Modica, "Effects of fish farming waste to sedimentary and particulate organic matter in a southern Mediterranean area (Gulf of Castellammare, Sicily): a multiple stable isotope study (δ13C and δ15N)," *Aquaculture,* vol. 234, pp. 199-213, 2004.

[128] R. Atanasova, L. Hadajinikolova and L. Nikolova, "Investigations on the biochemical composition of carp fish (Cyprinidae) blood al conditions of organic acuaculture," *Bulgarian Journal of Agricultural Science,* vol. 14, no. 2, pp. 117-120, 2008.

[129] S. Magill, H. Thetmeyer and C. Cromey, "Settling velocity of faecal pellets of gilthead sea bream (Sparus aurata L.) and sea bass (Dicentrarchus labrax L.) and sensitivity analysis using measured data in a deposition model," *Aquaculture,* vol. 251, pp. 295-305, 2006.

[130] M. Botero, "Comportamiento de los peces en la búsqueda y la captura del alimento," *Revista Colombiana de Ciencias Pecuarias,* vol. 17, no. 1, pp. 63-75, 2004.

[131] M. C. M. Beveridge, Cage Aquaculture, Oxford, UK: Blackwell Publishing Ltd., 2007.

[132] R. Pepe-Victoriano and R. Wilson, "Mejoramiento de las Condiciones de Cultivo en reproductores de Turbot (Scophthalmus maximus L.) en el Norte de Chile, para una mayor Producción de Huevos y Larvas.," Universidad de Antofagasta, Chile, 2007.

[133] A. G. V. Salvanes and P. J. B. Hart, "Individual variability in state-dependent feeding behaviour in three-spined sticklebacks," *Animal Behaviour,* vol. 55, no. 5, pp. 1349-1359, 1998.

[134] F. de la Gándara, «Efecto de Diversos Factores sobre el Consumo de Oxígeno de Juveniles de Seriola (Seriola Dumerili Risso, 1810) en Condiciones de Cultivo,» Universidad de Murcia, Murcia, Spain, 2003.

[135] E. Natalizio and V. Loscri, "Controlled mobility in mobile sensor networks: advantages, issues and challenges," *Telecommunications Systems,* pp. 1-8, 2001.

[136] Y. Liu and X. Ge, "Underwater Blue-Green Laser Sensor Network: Challenges and Approaches," *WSEAS Transactions on Communications,* vol. 5, no. 6, pp. 421-425, 2006.

[137] Y. Jeong, S. Shin, S. Park and C. Kim, "PBA: A New MAC Mechanism for Efficient Wireless Communication in Underwater Acoustic Sensor Network," *WSEAS Transactions on Communications,* vol. 6, no. 3, pp. 401-407, 2007.

[138] JTelectric, «Centralised Automatic Feed System,» JTelectric, 5 11 2012. [En línea]. Available: http://www.jtelectric.com/00414/00690/. [Último acceso: 5 11 2012].

[139] GNU, «GNU Octave,» GNU, 6 11 2012. [En línea]. Available: http://www.gnu.org/software/octave/index.html. [Último acceso: 6 11 2012].

[140] C. F. Pasluosta, Getting My Ph.D. Done!, 1 ed., vol. 1, Ruston, LA: Argentina Publications, 2010, pp. 20-25.

[141] J. M. Lopez, M. G. Watson and J. M. Fontana, "Writing Dissertations in New Word Formats," *LaTech Ph.D. Program,* vol. 1, no. 1, pp. 1-22, 08 07 2010.

[142] A. T. Hun, The Book of Irrelevant Citations, Ruston: Psychodelic Publishing Company, 2010.

[143] J. Michaelstein, "Equation Editor," *Microsoft Word 2010, The official blog of the Microsoft OfficeWord Product Team,* pp. http://blogs.msdn.com/b/microsoft_office_word/archive/2006/10/20/equation-numbering.aspx, 2006.

[144] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin and J. Stankovic, "ALARM-NET: Wireless Sensor Networks for Assisted-Living and Residential Monitoring," University of Virginia, Virginia, USA, 2006.

[145] H. Kim, B. Jarochowski and D. Ryu, "A Proposal for a Home-Based Health Monitoring System for the Elderly or Disabled," *Lecture Notes in Computer Science,* vol. 4061/2006, pp. 473-479, 2006.

[146] IEEE, "IEEE Std 802.11," IEEE Computer Society, NY, USA, 2007.

[147] IEEE, "IEEE Std 802.15.1," IEEE Computer Society, NY, USA, 2005.

[148] IEEE, "IEEE Std 802.15.4," IEEE Computer Society, NY, USA, 2006.

[149] ZigBee Alliance, "Zigbee specification," ZigBee Alliance, CA, USA, 2005.

[150] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad-Hoc Networks,* vol. 3, no. 3, pp. 325-349, 2005.

[151] C. M. Caffrey, O. Chevalerias, C. O'Mathuna and K. Twomey, "Swallawable-Capsule Technology," *IEEE Pervasive Computing,* vol. 7, no. 1, pp. 23-29, 2008.

[152] K. Finkenzeller, RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification, 2nd ed., West Sussex: Wiley, 2003.

[153] J. A. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter and B. Heile, "IEEE 802.15.4: A Developing Standard for Low-Power Low-Cost Wireless Personal Area Networks," *IEEE Network,* vol. 15, pp. 12-19, 2001.

[154] B. S. Hwang, J. M. Choi and K. S. Park, "A novel method for unobstrusive measurement of indoor activities using sensor-based monitoring system," in *International Special Topic Conference on Information Technology in Biomedicine*, Ioannina - Epirus, Greece, 2006.

[155] Y. G. Iyer, S. Gandham and S. Venkatesan, "STCP: a generic transport layer protocol for wireless sensor networks," in *14th International Conference on Computer Communications and Networks*, San Diego, CA, USA, 2005.

[156] B. P. Jarochowski, S. Shin, D. Ryu and H. Kim, "Ubiquitous Rehabilitation Center: An Implementation of a Wireless Sensor Network Based Rehabilitation Management System," in *International Conference on Convergence Information Technology, 2007.*, Gyeongju, Republic of Korea, 2007.

[157] V. Raghunathan, C. Schurgers, S. Park and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine,* vol. 19, no. 2, pp. 40-50, 2002.

[158] L. Schwiebert, S. K. Gupta and J. Weinmann, "Research challenges in wireless networks of biomedical sensors," in *7th annual international conference on Mobile computing and networking*, Rome, Italy, 2001.

[159] M. A. Vieira, C. N. Coelho, D. C. da Silva and J. M. da Mata, "Survey on wireless sensor network devices," in *IEEE Conference Emerging Technologies and Factory Automation*, Lisbon, Portugal, 2003.

[160] C. Wang, K. Sohraby, B. Li, M. Daneshmand and Y. Hu, "A survey of transport protocols for wireless sensor networks," *IEEE Network,* vol. 20, no. 3, pp. 34-40, 2006.

[161] Y. Wei, J. Heidemann and D. Estrin, "An energy-efficient MAC protocol for wireless sensor networks," in *IEEE Twenty-First Annual Joint Conference of the IEEE*

Computer and Communications Societies, New York, USA, 2002.

[162] M. O. Farooq and T. Kunz, "Operating Systems for Wireless Sensor Networks: A Survey," *Sensors,* vol. 11, no. 6, pp. 5900-5930, 2011.

[163] F. Brunetti, J. C. Moreno, A. F. Ruiz, E. Rocon and J. L. Pons, "A new platform based on IEEE802.15.4 wireless inertial sensors for motion caption and assesment," in *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, New York, USA, 2006.

[164] H. A. Yanco, "Wheelesley: A robotic wheelchair system: Indoor navigation and user interface," *Lecture Notes in Computer Science,* vol. 1458/1998, pp. 256-268, 1998.

[165] R. W. Gundersen, S. J. Smith and B. A. Abbott, "Applications of virtual reality technology to wheelchair remote steering systems," in *1st European Conference on Disability, Virtual Reality and Associated Technologies*, Maidenhead, UK, 1996.

[166] V. Santos, P. Bartolomeu, J. Fonseca y A. Mota, «B-Live-A Home Automation System for Disabled and Elderly People,» de *IEEE Second International Symposium on Industrial Embedded Systems*, Lisbon, Portugal, 2007.

[167] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk and J. Anderson, "Wireless sensor networks for habitat monitoring," in *1st ACM international workshop on Wireless sensor networks and applications*, Atlanta, Georgia, USA, 2002.

[168] E. Skafidas, "Precision Agriculture Information Networks: NICTOR Sensor Network Platform," 15 Augost 2012. [Online]. Available: http://www.cse.unsw.edu.au/~sensar/hardware/pictures/Nictor.pdf. [Accessed 15 Augost 2012].

[169] D. Hughes, P. Greenwood, G. Blair, G. Coulson, F. Pappenberger, P. Smith and K. Beven, "An intelligent and adaptable grid-based flood monitoring and warning system," in *UK eScience All Hands Meeting*, Nottingham, UK, 2006.

[170] J. Lloret, D. Bri, M. Garcia and P. Mauri, "A content distribution network deployment over WLANs for fire detection in rural environments," in *3rd international workshop on Use of P2P, grid and agents for the development of content networks*, Boston, Massachussets, USA, 2008.

[171] S. A. Summers, "Wireless Sensor Networks for Fire fighting and Fire Investigation," UCCS University of Colorado at Colorado Springs, CS526 Project, Berkley, CO, USA, 2006.

[172] G. Manes, R. Fantacci, F. Chiti, M. Ciabatti, G. Collodi, D. Di Palma and A. Manes, "Enhanced System Design Solutions for Wireless Sensor Networks applied to Distributed Environmental Monitoring," in *32nd IEEE Conference on Local*

Computer Networks, Dublin, Ireland, 2007.

[173] Alertsystems LTD, «Alertsystems LTD,» 1 January 2009. [En línea]. Available: http://www.alertsystems.co.uk/. [Último acceso: 15 August 2012].

[174] P. Bonnet, J. Gehrke and P. Seshadri, "Towards Sensor Database Systems," in *2nd International Conference on Mobile Data Management*, Hong Kong, China, 2001.

[175] Ericsson Microwave Systems, «C4ISR for Network Oriented Defense,» Ericsson Press, Stockholm, Sweden, 2006.

[176] M. Winkler, K. D. Tuchs, K. Hughes and G. Barclay, "Theoretical and practical aspects of military wireless sensor networks," *Journal of Telecommunications and Information Technology,* vol. 2/2008, pp. 37-45, 2008.

[177] K. Sohraby, D. Minoli and Z. Taieb, Wireless sensor networks, Technology, Protocols and Applications, Hoboken, New Jersey. USA: Wiley Interscience publication, 2007.

[178] H. Song, S. Zhu and G. Cao, "SVATS: A Sensor-Network-Based Vehicle Antitheft System," in *27th Conference on Computer Communications*, Phoenix, AZ, USA, 2008.

[179] C. Sharp, S. Schaffert, A. Woo, N. Sastry, C. Karlof and D. Culler, "Design and implementation of a sensor network system for vehicle tracking and autonomous interception," in *2nd European Workshop on Wireless Sensor Networks*, Istanbul, Turkey, 2005.

[180] J. Burrell, T. Brooke and R. Beckwith, "Vineyard Computing: Sensor Networks in Agricultural Production," *IEEE Pervasive Computing,* vol. 3, no. 1, pp. 38-45, 2004.

[181] H. Park, J. Burke and M. B. Srivastava, "Design and implementation of a wireless sensor network for intelligent light control," in *6th international conference on Information processing in sensor networks*, Cambridge, Massachusetts, USA, 2007.

[182] S. Kim, S. Pakzad, D. E. Culler, J. Demmel, G. Fenves, S. Glaser and M. Turon, "Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks," in *6th International Conference on Information Processing in Sensor Networks*, Cambridge, MA, USA, 2007.

[183] B. Stroustrup, The C++ Programming Language, 3rd ed., Boston: Addison-Wesley Longman Publishing Co., Inc., 1997.